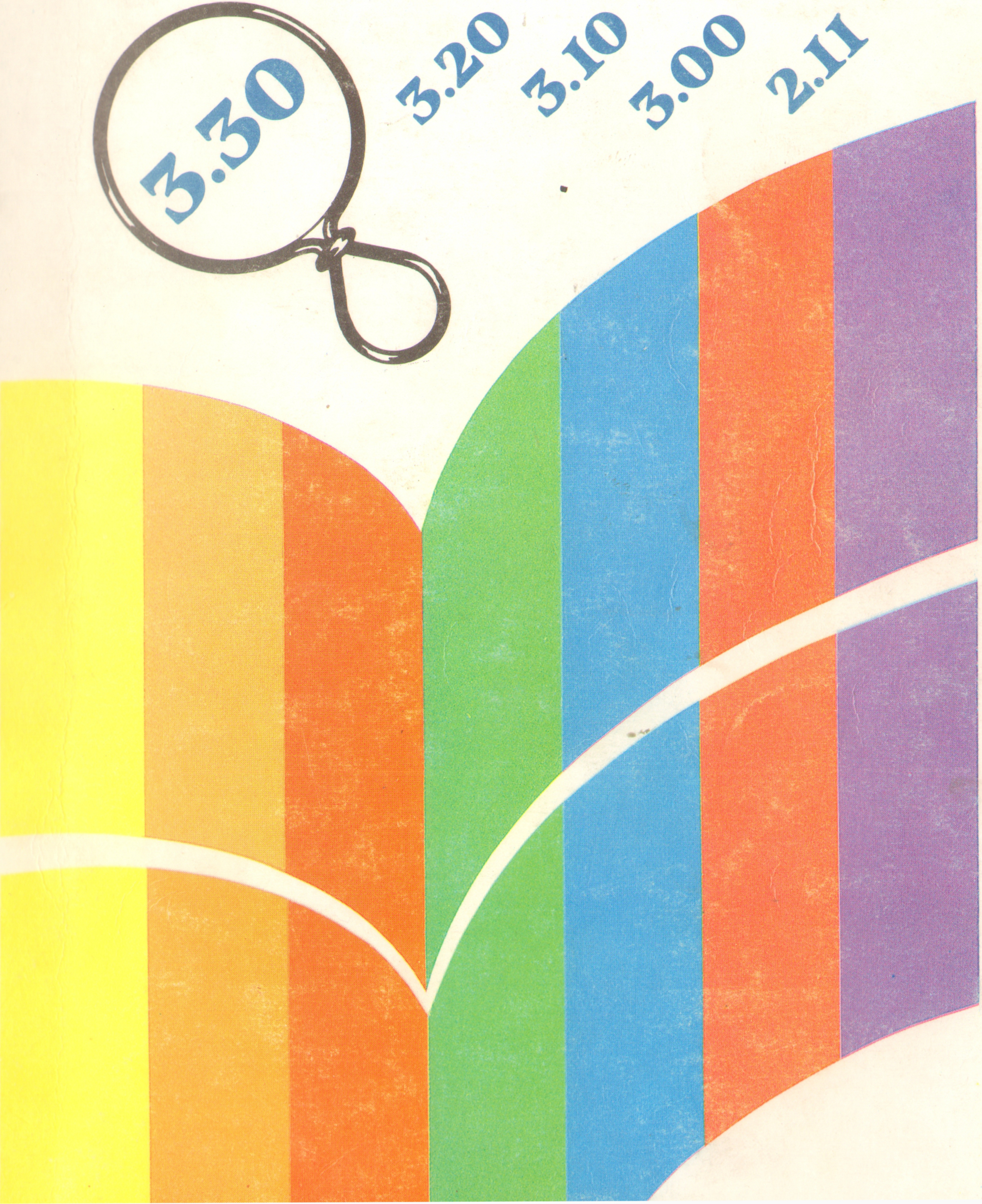
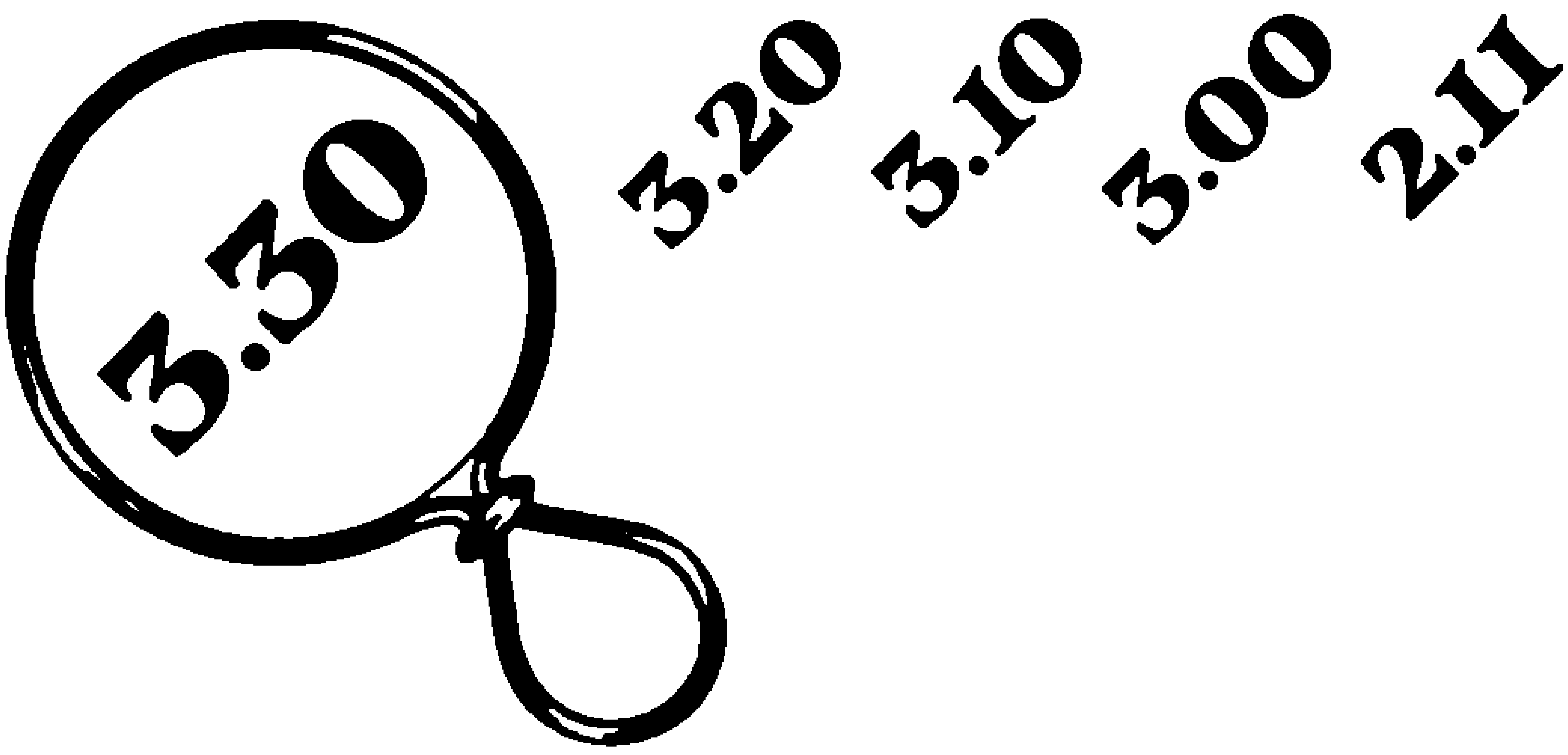
IBM PC DOS

I. A PC DOS HASZNÁLATA


3.30
3.20
3.10
3.00
2.11

IBM PC DOS

I. A PC DOS HASZNÁLATA


3.30
3.20
3.10
3.00
2.11

ALKALMAZÁSTECHNIKAI TANÁCSADÓ SZOLGÁLAT
BUDAPEST, 1990

Vörös Zsolt

Szerkesztette:	Szenes Katalin I-II. Dr. Úry László III.	
Készítette:	Bernáth István	1, 2, 8 és 9. fejezet
	Darvas László	3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16 és 17. fejezet
	Orosz Judit	10. fejezet
	Seres József	3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16, és 17. fejezet
	Úry László	3.xx kiegészítések
III. kötet:	Orosz Judit Baki Zoltán Pajor Gábor	
Lektorálta:	Éltető László Szenes Katalin Darvas László Donát János	
Munkatársak:	Szegedi Gyöngyi Kökényesi Judit	

Kiadó: LSI Alkalmazástechnikai Tanácsadó Szolgálat
Felelős kiadó: dr. Kovács Magda
Témafelelős: Székely László
Szedés: LSI ATSz Info Osztály
Technikai szerkesztő: Nagy Olivér
Engedélyszám: 57152
ISBN összkötet: 963 576 004/3
ISBN I. kötet: 963 576 0005/1
AGRO - PRINT Kft, Gyál 90-67/1
Felelős vezető: Tóth Antal

Előszó

Az IBM PC számítógépekre készült PC DOS operációs rendszer filozófiája híven tükrözi, hogy személyi számítógépekre (PC - Personal Computer) készült, azaz a gép előtt ülő, azt interaktívan kezelni akaró programozó számára. Lehetőségeit a felhasználó elsősorban közvetlenül kiadható parancsokkal éri el. A rendszert és az IBM PC assembler-t mélyebben ismerők számára pedig úgynevezett DOS funkciók állnak rendelkezésre, a software feladatok megvalósítása érdekében.

Ennyiben nagyon hasonlít egyik elődjéhez, a CP/M - szintén mikroszámítógépes - operációs rendszerhez, de annál funkció-állománya jóval bővebb, nagy interrupt-készlethez biztosít hozzáférést és ami a legfontosabb, a file-okat bonyolult fa-hierarchiákba lehet rendezni, ez pedig nemcsak a felhasználók, hanem az egyes feladatok file-halmazainak kényelmes, némi biztonságot is adó strukturálását is lehetővé teszi.

Könyvünket egyaránt szánjuk az IBM PC-vel most ismerkedő (bár lehetőleg egy kevés számítástechnikai alapismerettel már rendelkező), és a már tapasztalt programozóknak és rendszerprogramozóknak.

A DOS egyes változatai új verziószámot kapnak, amikor a rendszerek lényeges új vonásokkal egészülnek ki. Ezek közül a 2.10, 3.00, 3.10, 3.20 és a 3.30 verziókkal foglalkozunk.

A DOS indítása, file-, fix (Winchester) és cserélhető lemezekre vonatkozó parancsainak ismertetése után bemutatjuk a parancsok file-okban való tárolását és a parancs-file-ok indítását.

A PC DOS változatokkal az EDLIN nevű sorszerkesztő program áll a rendszerrel egyidőben a felhasználó rendelkezésére. Ennek parancsaival részletesen foglalkozunk.

Assembler programozáshoz szükséges nyomkövető-hibakereső program a DEBUG.

Futtatható programot a programszerkesztővel (konszolidátornak, linkernek is nevezik más gépeken) állíthatunk elő, ez a LINK program. Ez lehetővé teszi eredetileg különféle forrásnyelven és assemblerben készült programok összeépítését, és programkönyvtár file-ok felhasználását is.

A kezdők számára - a DEBUG-ról szólók kivételével - első lépésként ezen információk elsajátítását ajánljuk.

A fentiek után a rendszer konfigurálásával, a DOS speciális lehetőségeivel, belső felépítésével, a fix lemezen létrehozható fa-strukturájú file-szerkezettel, az operációs rendszer munkaterületeivel és vezérlő blokkjaival foglalkozunk.

Részletesen tárgyaljuk a DOS különböző verzióinál rendelkezésre álló funkciókat és azokat az interrupt-okat (megszakításokat), amelyekhez a felhasználó is hozzáfér, természetesen assembler szinten.

Egyes DOS parancsok ismertetésénél kitérünk a megfelelő PROPOS operációs rendszer szempontjaira is.

A függelék a rendszer hibaüzenetein kívül hasznos listákat (parancsok, alparancsok) és táblázatokat (file-név kiterjesztés, eszköznevek) tartalmaz.

Az index (tárgymutató) a fontosabb fogalmak főbb és magyarázó lelőhelyeit közli.

Az irodalomjegyzékben megadjuk azokat a könyveket, amelyek információit saját tapasztalatunkon kívül itt felhasználtuk.

Természetesen nem oldható meg az összes könyv teljes tudásanyagának közreadása egy ilyen kézikönyvben, amely egy kicsit tankönyv is, de reméljük, hogy a legtöbb, még rendszerprogramozási szintű feladat megoldásához is elég segítséget adunk.

Az alábbiakban felsoroljuk az LSI ATSz egyéb, IBM PC alkalmazóknak szóló kiadványait. A könyvek megrendelhetők az LSI ATSz címén. (Budapest 1300, Pf. 114.)

Megjelent:

Erdős Iván – Rusznák László: **IBM PC felhasználói és technikai információs kártya**

Dr. Pajor Gábor: **IBM PC-ről kezdő felhasználóknak I. A hardver; II. A szoftver**

Kápolnai A. – Geszti P.: **IBM PC BASIC fejlesztőrendszer**

Lipovszki György: **IBM PC FORTH**

Ráth György: **Programvédelmi rendszerek IBM PC-re**

Pirkó József: **3D perspektívikus grafika Turbo Pascal-ban IBM PC-n**

Dr. Úry László: **SYMPHONY I.**

Dr. Barakonyi Károly: **Táblázatkezelő rendszerek IBM PC-re**
(Lotus 1-2-3, Framework, Calcstar)

REXLIB
(Képernyőkezelő, file-kezelő rezidens rutin- és utility-csomag programozói kézikönyve Turbo Pascal programnyelven fejlesztők számára MS-DOS és NOVELL hálózati környezetben. Ára: 205.-Ft)

Előkészületben:

Dr. Úry László: **SYMPHONY II.**
(Várható megjelenés: 1988. december)

Pirkó József: **Turbo Pascal kezdőknek** (4.0 verzióig IBM PC-re)
(Várható megjelenés: 1989. I. negyedév)

Böér - Dóra - Fenyő - Seres: **Az IBM PC belső felépítése**
(Várható megjelenés: 1989. I. negyedév)

Mondovics János – Soltész Erzsébet: **dBASE III PLUS, CLIPPER**
(Várható megjelenés: 1989. I. negyedév)

Szenes Katalin

TARTALOMJEGYZÉK

I. kötet

Előszó 3
Tartalomjegyzék 5

1. FEJEZET

Bevezetés 11
1.1 Mi a PC DOS? 11
1.2 A PC DOS futtatásának hardware feltételei 13
1.3 A PC DOS verziói 15

2. FEJEZET

PC DOS alapismeretek 17

2.1 A PC DOS funkcionális részei 18
2.2 A PC DOS lemez-kezelése 18
2.3 A PC DOS indítása 21
  2.3.1 A DOS elindítása 21
  2.3.2 Ismételt indítás (restart) 23
  2.3.3 Egyéb tennivalók 24
  2.3.4 Fix lemezegység használata 25
2.4 A billentyűzet kezelése, DOS parancsok kiadása 27
2.5 Lemez és file kezelés 33
  2.5.1 A fastruktúrájú tartalomjegyzék 33
  2.5.2 File- és tartalomjegyzék nevek 35
  2.5.3 Aktuális lemezegység 38
  2.5.4 Keresési sorrend 38
2.6. A parancsok szintaxisa és a leírásuknál alkalmazott jelölésrendszer 39

3. FEJEZET

DOS: file-kezelő parancsok 41

3.1 COPY - másolás 41
3.2 XCOPY - file-ok csoportos másolása 45
3.3 REPLACE - file-ok cseréje 47
3.4 COMP - összehasonlítás 48
3.5 ERASE (DEL) - törlés 51
3.6 RENAME - átnevezés 52
3.7 RECOVER - helyreállítás 53
3.8 ATTRIB - file attribútumainak beállítása 54
3.9 SHARE - file megosztási környezet kezelése 55
3.10 TYPE - kiíratás 56
3.11 PRINT - kinyomtatás 57
3.12 FASTOPEN - file-ok elérésének gyorsítása 59

4. FEJEZET
DOS: lemezkezelési parancsok 60

4.1 MKDIR - altartalomjegyzék létrehozása 60
4.2 CHDIR - aktuális tartalomjegyzék kijelölése 61
4.3 RMDIR - altartalomjegyzék törlése 62
4.4 TREE - tartalomjegyzék struktúrájának kiíratása 63
4.5 PATH - bejárási utak kijelölése programokra 63
4.6 FORMAT - lemez formázása 64
4.7 SELECT - új rendszerlemez előállítása adott billentyűzet rutinnal, dátum és idő formátummal 68
4.8 SYS - rendszer-file-ok másolása 69
4.9 DIR - tartalomjegyzék kiíratása 70
4.10 VOL - kötetazonosító kiíratása 71
4.11 LABEL kötetazonosító megadása 72
4.12 DISKCOPY - lemez másolása 72
4.13 DISKCOMP - lemezek összehasonlítása 74
4.14 CHKDSK - lemez vizsgálata 75
4.15 VERIFY - írás utáni ellenőrzés 77
4.16 SUBST tartalomjegyzék helyettesítése lemezegység betűjelével 77
4.17 JOIN - lemezegység és tartalomjegyzék összekapcsolása 78
4.18 APPEND - bejárási utak kijelölése tetszőleges file-hoz 79

5. FEJEZET
A fix lemez kezelése 81

5.1 FDISK - fix lemez kialakítása 81
  5.1.1 DOS-partíciók előállítása 82
  5.1.2 Aktív partíció változtatása 84
  5.1.3 DOS-partíció törlése 84
  5.1.4 A partíció adatok kiíratása 85
  5.1.5 A következő fix lemez kiválasztása 85
5.2 Fix lemez archiválása 85
  5.2.1 BACKUP - file-ok archiválása 85
  5.2.2 RESTORE - file-ok visszatöltése 88

6. FEJEZET
A rendszer működését befolyásoló parancsok és kapcsolók 90
6.1 COMMAND - újabb parancsfeldolgozó program elindítása 90
6.2 EXIT - visszatérés az elsődleges parancsfeldolgozóhoz 91
6.3 CLS - képernyő törlése 92
6.4 DATE - dátum megadása 92
6.5 TIME - idő megadása 93
6.6 VER - DOS verzió lekérdezése 94
6.7 ASSIGN - periféria-hozzárendelések megváltoztatása 94

6.8 BREAK - a < Break > billentyű figyelése 95
6.9 MODE - perifériák működésmódjának beállítása 96
6.10 KEYBxx - nemzeti billentyűzet kiválasztása 101
6.11 GRAFTABL - grafikai jelek betöltése 102
6.12 GRAPHICS - grafikus nyomtatás lehetősége 103

7. FEJEZET

A parancs-file-ok 105

7.1 A parancs-file-ok meghívása és paraméterezése 105
  7.1.1 Formális paraméterek 106
  7.1.2 Környezeti változók használata 106
7.2 Az AUTOEXEC.BAT file 107
7.3 DOS alparancsok 108
  7.3.1 REM - megjegyzés 108
  7.3.2 ECHO - parancsok megjelenítése 108
  7.3.3 PAUSE - feldolgozás megszakítása 109
  7.3.4 IF - feltételes végrehajtás 110
  7.3.5 GOTO - ugrás a parancs-file-ban 112
  7.3.6 FOR - DOS parancsok ismételt végrehajtása 112
  7.3.7 SHIFT - a paraméter léptetése 113
  7.3.8 CALL - parancs-file hívása 114

8. FEJEZET

A sorszerkesztő program (EDLIN) 115

8.1 Az EDLIN indítása 116
8.2 Az EDLIN parancsai 117
  8.2.1 I (insert) - sorok beszúrása 119
  8.2.2 L (list) - sorok listázása 120
  8.2.3 P (page) - sorok listázása laponként 122
  8.2.4 Sorszám - sor szerkesztése 123
  8.2.5 D (delete) - sorok törlése 125
  8.2.6 E (end) - szerkesztés befejezése tárolással 128
  8.2.7 Q (quit) -kilépés a szerkesztésből 128
  8.2.8 M (move) - sorok áthelyezése 129
  8.2.9 C (copy) - sorok másolása 131
  8.2.10 S (search) - karaktersorozat keresése 132
  8.2.11 R (replace) - karaktersorozat helyettesítése 135
  8.2.12 T (transfer) - sorok beszúrása lemezről 137
  8.2.13 W (write) - sorok kiírása lemezre 138
  8.2.14 A (append) - sorok hozzáfűzése lemezről 139

9. FEJEZET

A nyomkövető-hibakereső program (DEBUG) 141

9.1 A DEBUG indítása 142
9.2 A DEBUG parancsai 143
  9.2.1 D (dump) - memóriatartalom kiíratása 145
  9.2.2 R (register) - regiszterek kiíratása, megváltoztatása 146
  9.2.3 F (fill) - memória feltöltése 147
  9.2.4 E (enter) - memóriatartalom megváltoztatása 148
  9.2.5 C (compare) - memóriaterületek összehasonlítása 149
  9.2.6 M (move) - memóriaterület átmásolása 150
  9.2.7 S (search) - keresés a memóriában 150
  9.2.8 Q (quit) - kilépés a DEBUG-ból 151
  9.2.9 A (assemble) - assemblálás 151
  9.2.10 U (unassemble) - disassemblálás 153
  9.2.11 G (go) - program futtatása 154
  9.2.12 T (trace) - lépésenkénti végrehajtás 155
  9.2.13 P - utasítássorozat végrehajtása 155
  9.2.14 N (name) - a file kijelölése 156
  9.2.15 L (load) - file (kijelölt szektorok) betöltése 157
  9.2.16 W (write) - file (kijelölt szektorok) kiírása 158
  9.2.17 H - hexadecimális műveletvégzés 160
  9.2.18 I - input a kijelölt port-ról 160
  9.2.19 O - output a kijelölt port-ra 160

10. FEJEZET

Futtatható program előállítása - programszerkesztés 161

10.1 A LINK indítása 161
  10.1.1 Programszerkesztés PC DOS parancs-sorral 161
  10.1.2 Programszerkesztés interaktív üzemmódban 162
  10.1.3 Programszerkesztés előre összeállított válaszfile-lal 164
  10.1.4 Könyvtárak elérési útjainak megadása 165
  10.1.5 A map-file szerkezete 166
  10.1.6 Ideiglenes lemez-file - VM.TMP 167
10.2 A LINK opciói 167
  10.2.1·Megszakítás a szerkesztés alatt - PAUSE 167
  10.2.2 Public szimbólumok táblájának előállítása 168
  10.2.3 A verem-méret meghatározása - STACK 168
  10.2.4 A kezdőcím maximumra állítása - HIGH 169
  10.2.5 Helyfoglalás az adatcsoportnak - DSALLOCATION 169
  10.2 6 Forrás-sorok kezdőcímének kiírása - LINE 170
  10.2.7 Szegmensszám beállítása - X 170
  10.2.8 'Régi' fordítók használata 170

11. FEJEZET

A rendszer konfigurálása 171

11.1 A konfigurációs file kezelése 171
11.2 A konfigurációs parancsok 171
  11.2.1 BREAK - a Break billentyű figyelése 171
  11.2.2 BUFFERS - lemez-pufferek száma 172
  11.2.3 COUNTRY - dátum, idő és egyéb formátum megadása 173
  11.2.4 DEVICE - periféria-kezelő rutinok 174
    - A kibővített képernyő- és billentyűzetkezelés (ANSI.SYS) 177
    - DISPLAY.SYS - kód-lapok használatának engedélyezése a képernyőn 178
    - A DIVER.SYS használata 179
    - PRINTER.SYS - kód-lapok használatának engedélyezése a nyomtatóra 180
    - VDISK.SYS - memórialemez használata 181
  11.2.5 FCBS - maximálisan megnyitható FCB-k száma 182
  11.2.6 FILES - egyidejűleg megnyitható file-o száma 182
  11.2.7 LASTDRIVE - a legnagyobb lemezegység megadása 182
  11.2.8 SHELL - felhasználói parancs-processzor 183
  11.2.9 STACKS - a DOS veremhasználatának módosítása 183

12. FEJEZET

A PC DOS speciális lehetőségei 184

12.1 A standard I/O átirányítása 184
12.2 A standard I/O összekapcsolása 185
12.3 Speciális DOS parancsok 186
  12.3.1 MORE - képernyő oldalakra tagolása 186
  12.3.2 FIND - keresés, válogatás 186
  12.3.3 SORT - rendezés 187
  12.3.4 PROMPT - a prompt karakter megváltoztatása 189
  12.3.5 CTTY - alternatív konzol kijelölése 190
  12.3.6 EXE2BIN - átalakítás EXE-ről COM formátumra 191
  12.3.7 SET - programkörnyezet kezelése 191
  12.3.8 NLSFUNC - ország-függő információk kezelése 193
  12.3.9 CHCP - kódlap-váltás valamennyi eszközre 193

II. kötet

13. FEJEZET

A PC DOS belső struktúrája

14. FEJEZET

Lemezkezelés és a felhasználói perifériakezelők szerkezete

15. FEJEZET

A DOS memóriakezelése, munkaterületei és vezérlőblokkjai

16. FEJEZET

DOS megszakítások

17. FEJEZET

DOS funkciók

FÜGGELÉKEK

A. A DOS üzenetei
B. Parancsok, alparancsok csoportosított betűrendes listája
C. Parancsok, alparancsok formája
D. File-név kiterjesztések
E. Eszköznevek listája
Tárgymutató
Irodalomjegyzék

1. Bevezetés

Ebben a fejezetben megismerkedünk a PC DOS operációs rendszer eredetével és fontosabb jellemzőivel, majd megvizsgáljuk, milyen hardware-rel rendelkező számítógép szükséges a PC DOS futtatásához. Végül áttekintjük a PC DOS elterjedtebb verzióinak (változatainak) specialitásait.

1.1 Mi a PC DOS?

Mielőtt rátérnénk a PC DOS leírására, tekintsük át röviden, mi a mikroszámítógépeknél az operációs rendszerek jelentősége, feladata. Az operációs rendszer maga is egy programrendszer, melynek programjai közvetítő szerepet töltenek be a felhasználó (és programjai), valamint a számítógép hardware erőforrásai között. Ezek a programok - többek között - a következő alapvető feladatokat látják el:

- programok betöltése a tárba és elindítása,
- input - output műveletek végzése (billentyűzet olvasása, kiírás a képernyőre ill. nyomtatóra, lemezek írása és olvasása, lemezes adatállományok létrehozása, másolása, törlése stb.),
- a lemezek tartalomjegyzékének (directory-jának) létrehozása, karbantartása és listázása,
- a billentyűzetről kiadott parancsok értelmezése és végrehajtása,
- a működés közben fellépett hibák kezelése.

Az a mód és sebesség, ahogyan az operációs rendszer e feladatokat végrehajtja, valamint a rendszer egyéb szolgáltatásainak bősége jelentősen befolyásolják számítógépünk viselkedését, meghatározzák azt a környezetet, amelyben a géppel dolgozhatunk. Az operációs rendszer minősége így gyakran fontosabb a felhasználó számára, mint a számítógép hardware tulajdonságai.

Az operációs rendszerrel akkor kerülünk közvetlen kapcsolatba, amikor a billentyűzetről parancsokat adunk ki, és azokat a rendszer végrehajtja (például file másolása, program elindítása).

Ne feledkezzünk meg azonban arról sem, hogy az általunk írt, vagy készen vásárolt programok is futásuk során számtalanszor igénybe veszik az operációs rendszer szolgáltatásait (többek között az input-output műveletek végrehajtásakor). Ebből következik, hogy a közvetlenül futtatható (gépi kódban írt, illetve arra lefordított) programok nemcsak hardware, hanem oprációs rendszer-függők is; működésük általában csak egy adott operációs rendszer keretein belül lehetséges.

Az egyszerűbb mikrogépekbe gyárilag beépítik (beégetik) az operációs rendszert. Nagyobb teljesítményű gépekhez azonban bonyolultabb rendszer készül, amelynek betöltése lemezről történik. Ekkor a felhasználó választhat, hogy az adott géphez kapható operációs rendszerek közül melyiket kívánja üzemeltetni. A PC DOS futtatására képes számítógépeken is általában mód van egyéb operációs rendszerek használatára (CP/M-86, VENIX, XENIX, QNX stb.).

A PC DOS operációs rendszert a Microsoft cég készítette az IBM PC számítógéphez, az IBM megrendelésére. A Microsoft cég megtartotta az operációs rendszer forgalmazói jogát, s a két cég egymástól függetlenül is végzett rajta fejlesztést. Ezért a Microsoft által önállóan forgalmazott MS DOS - s a klónokkal általában ezt a rendszert adják - bizonyos eltéréseket mutat az IBM PC DOS-tól. Leírásukkor az eltéréseket feltüntetjük.

A DOS szó a **Disk Operating System** (lemezes operációs rendszer) rövidítése. Az IBM gépei (PC, PC XT, PC AT, PPC, PS2 stb.), valamint más gyártók IBM kompatibilisnek nevezett termékei hatalmas sikert arattak a professzionális mikroszámítógépek piacán. Bár az IBM más operációs rendszereket is forgalmaz gépeihez, mégis a PC DOS csaknem egyeduralkodóvá vált ezek között. A siker néhány oka: alacsony ár, könnyű kezelhetőség, széleskörű szolgáltatások, az IBM messzemenő támogatása.

A PC DOS nem-multiprogramozható egyfelhasználós operációs rendszer. Mint a neve is mutatja, a billentyűzeten, a képernyőn és a nyomtatón kívül a lemezek - hajlékony- és fixlemezek (Winchesterek) - kezelésére is képes, sőt a rendszer más perifériák vezérlő rutinjaival is kibővíthető. Mi több, legalább egy lemezegységre szüksége van. A parancsokat a billentyűzetről és az ún. parancs-file-okból is elfogadja (interaktív, ill. kötegelt, batch üzemmód).

Lemez- és file-kezelésének főbb jellemzői:

- hierarchikus, fastruktúrájú tartalomjegyzék (directory),
- szekvenciális és közvetlen (direkt) hozzáférést támogató rutinok,
- dinamikus helykezelés a lemezeken (a file-ok mérete változhat),
- file-ok keresésekor globális file-nevek használata,
- egységhibák jelzése, kezelése,
- a programok standard inputjának (billentyűzet), ill. outputjának (képernyő) átirányítási lehetősége (pl.lemezről, ill. lemezre),
- két program összekapcsolása úgy, hogy az egyik program standard outputja a másik program standard inputja lesz (piping)

A PC DOS számos hasznos segédprogramot tartalmaz. A fontosabb segédprogramok funkciói:

- file-ok másolása, összehasonlítása, átnevezése, megjelenítése, nyomtatása,
- floppy lemezek formázása, másolása, összehasonlítása,
- fix lemezek archiválása floppy-ra és visszatöltése,
- szöveges file-ok létrehozása és átszerkesztése,
- gépi kódú programok nyomkövetése, hibakeresés,
- tárgy-(object) programok összeszerkesztése,
- file-ok válogatása, rendezése.

A PC DOS-hoz általában mellékelik a BASIC nyelv interpreterének egy vagy két változatát is. A BASIC programozással e könyvben nem foglalkozunk.

A PC DOS sok tulajdonságát a hagyományos CP/M rendszertől örökölte, de a UNIX operációs rendszer egyes lehetőségeit (hiearchikus tartalomjegyzék, input és output átirányítása, összekapcsolása, vagy a MORE parancs, bizonyos DOS funkciók) *is megvalósítja.*

Túl a felsoroltakon, a PC DOS legfontosabb jellemzője: népszerűsége és elterjedtsége. Ennek köszönhető ugyanis, hogy a legtöbb mikrogépgyártó PC DOS futtatására képes gépeket is gyárt, nagyon sok software-cég pedig mikrogépes programjait először a PC DOS-hoz készíti el. Ha tehát a felhasználó a PC DOS-t választja, biztos lehet benne, hogy a "fő áramlatban úszik", és még hosszú évekig a legszélesebb választékból szerezheti be a feladatai ellátásához szükséges hardware- és software-eszközöket. A saját fejlesztésű programokat is érdemes PC DOS alatt megírni, mert - ha a jövőben más, nagyobb teljesítményű gépre kell áttérnünk - akkor is biztosan választhatunk olyan számítógépet, amelyiken a PC DOS valamelyik változata működik, és így nem vész kárba a gépváltáskor a korábban befektett munka.

A PC DOS az IBM személyi számítógépével, az IBM PC-vel együtt fejlődött. Az eredeti gép még kazettás egység kezelésére is alkalmas volt. Jelenleg hazánkban legelterjedtebb a PC vagy MS DOS 3.10-es változata (1987.május 6-án!), a hazai forgalmazók is ezt szállítják. Azok azonban, akik megtehetik, már térnek át a jelenleg legfejlettebb 3.30-as változatra.

1.2 A PC DOS futtatásának hardware feltételei

A PC DOS futtatásához IBM PC/XT/AT vagy azzal kompatibilis mikroszámítógépre van szükség. Ezek Intel 8086-os típusú, vagy azzal kompatibilis utasításkészletű mikroprocesszorral rendelkeznek.(Néhány kompatibilis típus: 8088, 80286, 80386.) E processzorok jellemzői: tizenhat bites belső szóhossz, nyolc- tizenhat- vagy harminckét bites adatbusz, húsz-huszonnégy bites címbusz, négy-húsz Megahertz-es órajel. A fő processzor mellett a 8087-es (vagy 80287-es) aritmetikai segédprocesszor is alkalmazható, amely jelentősen felgyorsítja a lebegőpontos műveleteket.

Egyes géptípusokon < TURBO > feliratú gomb található, amely megnyomása az órajel sebességét 20-40%-kal megnöveli. Ilyen gépeket általában célszerű mindig „turbo" üzemmódban működtetni.

A számítógép operatív tára csak olvasható (ún.ROM), és írható-olvasható (ún.RAM) memóriából áll. A ROM tár mérete 64 Kbyte alatt van, és a következő funkciókat ellátó rutinokat tartalmazza:

- a számítógép bekapcsoláskori ellenőrzése (ön-teszt),
- input-output alapműveletek végrehajtása a gépbe beépített (beépíthető) perifériákra,
- a számítógép aktuális konfigurációjának felismerése,
- a képernyőn ábrázolható karakterek pontképe (grafikus módban),
- BASIC interpreter (nem minden géptípusnál).

Megjegyezzük, hogy az IBM legújabb, Personal System/2 sorozatában a ROM mérete 128 Kbyte.

A ROM-ba beégetett rutinokat a DOS is gyakran felhasználja, elsősorban a BIOS-nak nevezett rutingyűjteményt, amely az input-output műveleteket végzi. (A BIOS szó a Basic Input Output System input-output alaprendszer rövidítése.)

Az írható-olvasható (RAM) memória mérete általában 256 és 640 Kbyte között változik. *Utólag azonban a RAM tovább bővíthető; az elérhető maximum 640 Kbyte.* Az AT gépeknél további tárbővítés is lehetséges (3 Mbyte-ig), de a DOS az 1 Mbyte feletti tárat nem tudja kihasználni, legfeljebb "virtuális lemezként" (floppy lemez szimulációjára).

A PC DOS-t futtató gépek kötelező perifériái: billentyűzet, képernyő és legalább egy lemezmeghajtó egység. A billentyűzetek különböző betűkészlettel és elrendezésben készülnek, az egyes nemzeti nyelvek sajátosságai szerint. A képernyőt (monitort) a hozzátartozó vezérlő kártyával együtt az alábbi kategóriákba lehet sorolni:

- monokróm vagy színes,
- alfanumerikus vagy grafikus.

A monokróm monitorok csak kétféle színt biztosítanak (például zöldet és fehéret). Az alfanumerikus monitorok csak a gép jelkészletét képező 256 karakter ábrázolására képesek. A grafikus monitorok pontonként is címezhetők, bármelyik pontjuk önállóan is kezelhető (például egy 640-szer 200-as pontmátrixban). Szöveges megjelenítés esetén a legtöbb monitor 25 sorban 80-80 karaktert ír ki.

A következő főbb monitorok és csatolókártyák léteznek:

- monokróm (monokróm, alfanumerikus)
- Hercules monografikus (monokróm, grafikus, 720x350 képpont)
- színes grafikus (640x200 képpont, ff. ált. 320x200 képpont 4 színben)
- nagyfelbontású színes grafikus (640x350 képpont 16 színben)

A legismertebb csatoló kártyák:

MCA (monocrome adapter)
HGA (Hercules graphics adapter)
CGA (color graphics adapter)
EGA (enchanced graphics adapter)
PGA (professional graphics adapter)

Természetesen léteznek más monitortípusok is (pl. az IBM új PS/2 gépcsaládjánál, illetve tervezői - CAD - munkához). A legjobbak felbontása 2048x2048 képpont is lehet 4096 színben.

A lemezmeghajtó egységeket két nagy csoportra osztjuk: **a hajlékony (floppy) és a fix (Winchester) lemezeket kezelő egységekre.** A floppy lemezek cserélhetők,egy- vagy kétoldalasak, tárolási kapacitásuk általában 160-tól 1440 Kbyte-ig terjed. A fix lemezek a meghajtó egységükkel együtt be vannak építve a számítógépbe, nem cserélhetők. Elérési sebességük igen nagy, tárolási kapacitásuk kb. 10-200 Mbyte.

A nem kötelező perifériák közül kiemeljük a **nyomtatókat** és az **RS-232 adaptert** (utóbbi a számítógépek egymással való összeköttetésére, illetve bizonyos perifériák csatlakoztatására szolgál). Elképzelhető - többek között - rajzgép (plotter), digitalizáló rajztábla (digitizing tablet), botkormány (joystick), potenciométer (paddle), egér (mouse), fényceruza (light pen), modem és különböző terminál-emulátor kártyák bekötése is a számítógépbe.

A nagykapacitású Winchesterek fokozatos terjedése növelte az igényt, hogy az azon levő adattartalom menthető legyen **cserélhető médiára** is. Jelenleg nagyon sokféle ún. **streamer-egység** vásárolható az IBM kompatibilis számítógépekhez. Egy-egy kazettára 10, 20, 40 illetve 60 Mbyte-nyi információ menthető ki. Ezekhez az egységekhez külön program tartozik, amelyet az egységhez adnak.

A PC DOS alatt üzemelő számítógépek hardware és software szempontból is igen rugalmas struktúrát alkotnak, ami megkönnyíti a gép hozzáigazítását a változó, növekvő feladatokhoz. **Bővíthető a RAM memória, és szabad kártyapozíciók vannak a gépben, amelyek különböző perifériák illesztésére is felhasználhatók.** A gépet más számítógépekkel össze lehet kapcsolni, és nagy- vagy minigépek intelligens termináljaként üzemeltetni.

A PC DOS is nyílt rendszert alkot, módosítására és bővítésére számos lehetőség kínálkozik (gondoljunk például az új perifériákat vezérlő rutinok beillesztésére).

Végül egy figyelmeztetés az "IBM kompatibilisként" hirdetett számítógépek vásárlói számára: mielőtt ilyen gépet vennénk, győződjünk meg arról (lehetőleg kipróbálás útján), hogy az adott konfiguráción valóban lefutnak-e az alkalmazni kívánt programok (beleértve a PC DOS-t is). Ha ez a feltétel teljesül, még akkor sem lehetünk benne biztosak,hogy más programok - amiket a későbbiekben vásárolnánk - szintén hibátlanul működnek a kiválasztott gépen. Ezért lehetőleg olyan számítógépet vegyünk, amelynek kompatibilitásáról (hasonlóságáról) komoly garancia vagy tapasztalat áll rendelkezésre. Akinek lehetősége van rá, próbáljon írásos szakvéleményt beszerezni a vásárolni kívánt konfiguráció kompatibilitási szintjéről. (DOS, BIOS vagy hardware kompatibilitás)

Ez a figyelmeztetés különösen érvényes a mikrogépes hálózatokra. Sajnos a hazai gyakorlat ezen a téren különösen rossz: házilag barkácsolt kártyák és licensz nélküli szoftverek tömege lepte el az országot. A hálózati hibák nehezen kiküszöbölhetők, s általában véletlenszerűen jelennek meg. A 100%-os kompatibilitás hálózati üzem esetén nélkülözhetetlen!

1.3 A PC DOS verziói

A PC DOS nincs fixen beépítve a számítógépekbe, betöltése lemezről történik, ezért a rendszer továbbfejlesztését, különböző géptípusokhoz való igazítását semmi nem gátolja. Az első IBM PC számítógépekhez eleinte a DOS 1.00, később az 1.10 verzióját árusították. Ezek a változatok még nem kezelték a fix (Winchester) lemezeket. Az IBM PC XT gépek már tartalmaznak fix lemezt, ehhez a géptípushoz dolgozta ki a Microsoft a 2.00 verziót. A DOS 2.00 verziója a fix lemez kezelésén kívül más újdonságot is tartalmaz, például a hierarchikus (fastruktúrájú) directory-t (tartalomjegyzéket), a standard input és output átirányítását vagy összekapcsolását, a háttérben történő nyomtatást, javított

BASIC interpretert stb. A DOS 2.11 verziója a hordozható IBM PC-n (PPC) is futtatható, egyébként felhasználói szempontból teljesen megegyezik a 2.00 verzióval.

Az IBM PC AT nagyobb kapacitású floppy és fix lemezeit csak a DOS 3.00 és későbbi verziói tudják kezelni. A 3.00 verzió a 2.00-hoz képest a következő lényeges továbbfejlesztéseket tartalmazza: virtuális lemezkezelés a RAM-ban, javított programszerkesztő (LINK) és BASIC interpreter, file-ok törlés elleni védelme, file-ok osztott használata (file sharing) stb. A 3.10 verzió pedig a számítógép hálózatba kötésekor futtatott hálózati software-t kiszolgáló rutinokat is tartalmazza, de egyébként megegyezik a 3.00-val. A DOS 3.20-as változata további új utasításokat tartalmaz. A DRIVER.SYS driver lehetővé teszi, hogy különböző típusú hajlékonylemezes egységeket tartalmazzon a rendszer. A régi utasítasítások közül hetet továbbfejlesztettek. A DISKCOMP és a DISKCOPY parancs a 720 Kbyte-os egységre is alkalmazható.

A FORMAT nem formázza meg az aktuális meghajtót, csak ha a lemezegység betűjelét beírjuk a parancsba. Ez bizonyos védelmet jelent a fix lemezegység újraformázása ellen.

Az IBM és Microsoft legújabb közös fejlesztése, a DOS 3.30-as verziója leglényegesebb újítása, hogy lehetővé teszi a 32 Mbyte-nál nagyobb fix lemezek (winchesterek) használatát is.

A felsorolt DOS verziók egymással felülről kompatibilisek, tehát a kisebb számú verzió alatt készült felhasználói programok a nagyobb verziószámú rendszerek alatt is futtathatók, így semmi nem gátolja a fejlettebb verziókra való áttérést.

Az IBM kompatibilis számítógépekhez a legkülönbözőbb DOS változatokat forgalmazzák, amelyek többé-kevésbé eltérnek a PC DOS említett verzióitól. Az IBM PC-khez legjobban hasonlító gépeken azonban a 2.11 vagy 3.00 verziók is lefutnak, ilyenkor érdemes ezeket alkalmazni.

Könyvünk írásakor el kellett döntenünk, hogy melyik DOS változattal foglalkozzunk. Választásunk az IBM által forgalmazott legfejlebb DOS 3.20-as verziójára esett. Könyvünk írásakor sajnos még nem állt rendelkezésre a 3.30-as verzió részletes leírása. A részletes - technikai - kézikönyv akkor még külföldön sem jelent meg. A második szerkesztés során már a 3.30-as újdonságokat is be tudtuk építeni a könyvbe.

Természetesen minden esetben feltüntetjük, hogy a korábbi verziók milyen formában használják a szóbanforgó parancsot. Könyvünk "visszafelé" az IBM PC DOS 2.10 változatig tartalmaz információt (beleértve az MS DOS 2.11, DOS 3.00, DOS 3.10 és DOS 3.20-as változatokat.)

Könyvünk kijelentései és példái tehát csak a DOS 3.30 alatt üzemelő gépekre érvényesek teljes mértékben. Sok IBM kompatibilis gép (pl. Commodore PC-10, PC-20, valamint a magyar PROPER-16 számítógép felhasználóinak kedvéért azonban kitérünk e gépek operációs rendszereinél (MS DOS 2.11, ill. a PROPER-16-nál a PROPOS 3.0) tapasztalható lényeges eltérésekre. Más DOS változatok futtatásakor további különbségekre kell számítanunk, melyek felderítéséhez az adott rendszer referencia-kézikönyvét is elő kell vennünk.

2. PC DOS alapismeretek

Ebben a fejezetben olyan alapkérdésekkel foglalkozunk, amelyek ismerete már az első DOS parancs kiadása előtt nélkülözhetetlen. Ezek a következők:

- a DOS funkcionális részei,
- a DOS lemezkezelése,
- a DOS indítása és újraindítása
- a billentyűzet kezelése DOS-ban,
- a parancsok leírásánál alkalmazott jelölések.

Az első két témát itt még nem tárgyaljuk teljes részletességgel, csak a továbbhaladáshoz szükséges mértékben. A következő fejezettől kezdve pedig már a DOS konkrét alkalmazásával, az egyes DOS parancsokkal foglalkozunk.

2.1 A PC DOS funkcionális részei

A DOS operációs rendszert tartalmazó floppy lemezeken számos program található. Két - a DOS szempontjából fontos - programot azonban nem láthatunk a tartalomjegyzékben, mert ezek rejtett (hidden) jelzéssel vannak ellátva. A rendszert az operatív tárba betöltő kis program pedig nem önálló file-ként helyezkedik el a lemezen, ezért file-neve sincsen. A DOS programjait funkciójuk szerint a következő csoportokba oszthatjuk:

- betöltő program (betölti a DOS-t a tárba),
- IBMBIO.COM program (a gép ROM tárába beégetett BIOS rutinokkal tartja a kapcsolatot, MS DOS neve: IO.SYS)
- IBMDOS.COM program (a felhasználói programokkal tartja a kapcsolatot, MS DOS neve: MSDOS.SYS).
- COMMAND.COM program (feldolgozza a DOS parancsokat),
- a DOS "külső" parancsait tartalmazó segédprogramok,
- a BASIC interpreter egy vagy két változata, mintaprogramokkal.

A lemezes input-output műveleteket számítógépünk a DOS közvetítésével hajtja végre.

Maga a DOS pedig a következőképpen kerül be a lemezről az operatív memóriába: a gép bekapcsolásakor, valamint a rendszer újraindításakor a ROM tárba fixen beépített program (amely független az operációs rendszerektől) beolvassa a lemez legelső szektorának tartalmát a RAM-ba, és végrehajtja azt. Az első szektorban éppen a betöltő program (boot record) helyezkedik el, amely – DOS lemezek esetén – betölti a tárba az IBMBIO és IBMDOS rendszerprogramokat.

Az IBMBIO.COM program "rejtett" jelzéssel van ellátva, a tartalomjegyzéken nem jelenik meg. (A PROPOS 16 operációs rendszerben: PROBIO.COM) Feladata a ROM-ban elhelyezkedő BIOS (input-output alapműveleteket végző) rutinokkal való kapcsolattartás. Ahogyan maga a BIOS, az IBMBIO program is erősen gépfüggő.

Az IBMDOS.COM program is "rejtett" jelzésű. (A PROPOS 16 operációs rendszerben: PRODOS.COM) Ez a program tartja a kapcsolatot a felhasználói programokkal, végrehajtja azok kívánságait, közvetít az IBMBIO felé. Az IBMBIO és az IBMDOS betöltésük után végig a gép memóriájában maradnak, csökkentve a felhasználói programok számára fennmaradó szabad területet.

A COMMAND.COM programot parancs-processzornak, parancs-feldolgozónak is nevezik. Ez olvassa be, értelmezi és hajtja végre a billentyűzetről (vagy a parancs-file-okból) az általunk kiadott DOS parancsokat. Az ún. belső parancsok kódját, programját is a COMMAND.COM tartalmazza, a külső parancsokat pedig a lemezen elhelyezkedő DOS segédprogramok futtatásával hajtja végre. A parancsprocesszor kisebb része állandóan a memóriában foglal helyet, de nagyobbik részét szükség esetén a felhasználói programok kitörölhetik, és tárterületét saját céljukra felhasználhatják. Utóbbi esetben a program lefutása után a COMMAND.COM-ot ismét be kell tölteni a rendszerlemezről (ezt a DOS, pontosabban a COMMAND.COM rezidens része automatikusan meg is teszi).

A gyári DOS lemezen található .COM és .EXE kiterjesztésű programokat nevezzük DOS segédprogramoknak, ill. külső parancsoknak. Említettük, hogy a legfontosabb funkciók (belső parancsok) kódját a parancsprocesszor tartalmazza, ezek végrehajtásához nem kell a lemezről programokat beolvasni. A ritkábban igényelt feladatok (külső parancsok) programját azonban nem lett volna érdemes a COMMAND.COM-ba beépíteni, nehogy annak mérete túl nagyra nőjön, és elvegye az operatív tárban a felhasználói programok elől a helyet. Ezért, amikor olyan DOS parancsot adunk ki, amelynek kódja nem szerepel a parancsprocesszorban, a DOS a lemezen keresi meg a parancs nevével megegyező, .COM vagy .EXE kiterjesztésű programot, és elindítja azt.

Ebből is látszik, hogy nincs lényeges különbség a DOS külső parancsai és az egyéb végrehajtható (gépi kódú) programok között; mindkét típusba tartozó programok a lemezen önálló file-ként helyezkednek el (.COM vagy .EXE kiterjesztéssel), és keresésük, lefuttatásuk azonos módon történik.

A DOS parancsok tárgyalásakor mindig megemlítjük majd, hogy belső vagy külső parancsról van-e szó.

Az említett DOS programokon kívül a rendszerlemezeken általában megtaláljuk a BASIC interpreter egy vagy két változatát is.

2.2 A PC DOS lemez-kezelése

A DOS verziói hajlékony (floppy) és fix (Winchester) lemezeket is tudnak kezelni. A DOS-t futtató számítógépek általában egy vagy két floppy-meghajtóval vannak felszerelve (5 1/4 vagy 3 1/2 inch átmérőjű lemezek számára). A fix lemezek használata nem kötelező, de nagyobb adattárolási igények esetén egy vagy két Winchester-meghajtóval is célszerű kiegészíteni a konfigurációt.

A floppy-meghajtók egy vagy két fejjel rendelkeznek, ettől függően a lemezeket csak az egyik vagy mindkét oldalukon tudják írni, illetve olvasni. Az adatok tárolása a floppy lemezeken koncentrikus körökben, ún. sávokban (track) történik. A sávok tovább oszlanak szektorokra. Egy-egy szektorban (a DOS alatt) 512 adat-byte tárolható. A sávok és szektorok száma meghajtó típusonként más és más lehet. A DOS a lemezeket negyven, egyes esetekben nyolcvan sávosra formázza, sávonként (választhatóan) nyolc vagy kilenc szektorral. A lemezek adattárolási kapacitását így a következő táblázat szemlélteti:

egyoldalas lemez, 40 sáv 8 szektor/sáv: 160 Kbyte
egyoldalas lemez, 40 sáv 9 szektor/sáv: 180 Kbyte
kétoldalas lemez, 40 sáv 8 szektor/sáv: 320 Kbyte
kétoldalas lemez, 40 sáv 9 szektor/sáv: 360 Kbyte
kétoldalas lemez, 80 sáv 9 szektor/sáv: 720 Kbyte,
kétoldalas,nagykapacitású lemez, 80 sáv 15 szektor/sáv: 1.2 Mbyte

Az egyfejes meghajtókat az újabb modellekben már nem alkalmazzák, a nyolcszektoros tárolás pedig csak a DOS korábbi verzióival fennmaradó kompatibilitás érdekében maradt fenn. A leggyakoribb kapacitás így a 360 Kbyte, illetve az AT gépeken az 1.2 Mbyte. A 720 Kbyte-os lemezegység támogatása elsősorban az egyéb rendszerekkel való adatcsere biztosítására szolgál.

(Fontos tudni, hogy az IBM a 80 sávos és 9 szektor/sáv típusú lemezegységeket csak a 3.10-es verziótól támogatja, elsősorban a külső egységek kezelésére. A PROPER gépek 80 sávos lemezformázási módja nem kompatibilis a DOS-éval, így ezeket a PC-DOS alatt nem lehet olvasni, illetve írni.)

Az IBM legújabb, PERSONAL SYSTEM/2 számítógépcsaládja már nem 5 1/4, hanem 3 1/2 inch-es floppy meghajtóegységeket tartalmaz. Ezek kétoldalas, 80 sávos, tehát 720 Kbyte tárolókapacitású lemezeket használnak. A PS/2 gépcsaládhoz külső egységként csatlakoztathatók 5 1/4 inch-es egységek.

Ha valaki 80 sávos egységét tartalmazó PROPER gépét át akarja állítani a PROPOS operációs rendszerről PC-DOS-ra, a következőt kell tennie:

1. Az eddigi 80 sávos PROPOS formájú lemezeinek tartalmát át kell másolnia 40 sávos (PC-DOS kompatibilis) lemezekre.
2. A fix lemezt el kell mentenie 40 sávos lemezekre.
3. A DOS 3.20 rendszerlemezről újraindítva (boot-olva) a rendszert, újra kell formáznia a fix lemezt.
4. A fix lemezre vissza kell tölteni a file-okat a mentési lemezről.

A DOS 3.00 verziója már az 1.2 Mbyte-os (1200 Kbyte-os) floppy-k kezelésére is képes, de más kapacitású meghajtók is forgalomban vannak.

Az egyes meghajtókon létrehozott lemezek nem mindig olvashatók be más típusú meghajtókon. *Az 1.2 Mbyte-os meghajtón például be lehet olvasni 360 Kbyte-os lemezeket, de fordítva ez már nem igaz.* Ügyelni kell arra is, hogy oldalszámban és tárolási sűrűségben megfelelő lemezeket vásároljunk az adott floppy-meghajtóhoz. **Speciálisan AT-ra készült lemezeket - azok mágneses tulajdonságai miatt - az XT-s gépeken nem lehet használni.**

Sajnos az 1.2 Mbyte-os egységek fizikailag eltérő felírási módja miatt hiába írunk fel XT-n megformázott 360 Kbyte-os lemezekre az AT-n, azt nem mindig tudjuk utána az XT-n elolvasni. Emiatt az XT-AT kapcsolat - kivéve a nagyon ritkán található igazán jól beállított AT-kat -többnyire egyoldalú: az AT tudja olvasni az XT-n felírt lemezeket, fordítva ritkaságnak számít.

A floppy lemezek oldalán egy kisebb bevágást találunk, erre ragasztható a lemezekhez mellékelt alumínium vagy papír írásvédő csík. A csík felragasztásával lehet a lemezt megvédeni törlés és felülírás ellen.

Ilyenkor a gép csak olvasni tudja az adott lemezt. A floppy-lemezek sajnos sérülékenyek, érzékenyek a hőre, porra, mágneses térre, kézzel történő érintésre. Ezért a teljes biztonság érdekében rendszeresen készítsünk másolatot a fontos programokról, adatokról.

A 3 1/2 inch-es floppy lemezek legfőbb erénye, hogy kevésbé sérülékenyek, mivel az egységből kivéve mágneses részük teljesen fedett. (Az író-olvasó rést egy csúszófedő védi, a lemezt a meghajtóba téve teszi csak szabaddá a rést.)

Az üres floppy-kat első felhasználásuk előtt a DOS megfelelő parancsával meg kell formáznunk. E művelet során alakítja ki a DOS a lemezen a sávokat és szektorokat (egyelőre üres adattartalommal). A formázáskor megadhatunk bizonyos opciókat, amelyek a létrehozandó sávok és szektorok számát befolyásolják. Előírhatjuk továbbá, hogy az új lemezre átmásolódjon-e maga a DOS rendszer.

A DOS betöltő programja (boot record) a formázáskor minden lemezre átkerül, de az IBMBIO és az IBMDOS programok csak abban az esetben, ha a formázó parancsban a DOS rendszerprogramok másolását is előírtuk. (A DOS külső parancsait tartalmazó segédprogramokat a formázás után kiadott másoló paranccsal lehet a lemezre vinni.) A DOS-t természetesen csak az említett három alapprogramot tartalmazó lemezekről lehet betölteni, elindítani. Ha a betöltő program nem találja az alapprogramokat, akkor hibajelzést ad. A rendszer elindítása után már nem kell, hogy fent legyen a DOS-t tartalmazó lemez.

A fix (Winchester) lemezek közös tengelyre szerelt merev korongokból állnak, melyek egybe vannak építve a meghajtóegységgel, tehát nem cserélhetők. Az egyes felületek - floppy-khoz hasonlóan - sávokra és szektorokra oszlanak. A szektorok itt is 512 adat-byte tárolására képesek. Minden felülethez külön író-olvasó fej tartozik. Az egymás fölött elhelyezkedő (fejmozgás nélkül elérhető) sávokat együttesen cilindereknek hívjuk. A lemezek szokásos kapacitása 10, 20, 40 Mbyte, de ennél nagyobb méretek is előfordulnak.

Felhívjuk a figyelmet arra a tényre, hogy a winchestereknek kétféle kapacitásuk létezik, egy formázatlan (nyers) és egy formázott. A leggyakrabban használt winchester típus 611 cilindert tartalmaz. Ennek formázatlan kapacitása 27 Mbyte, formázott kapacitása azonban csak valamivel több, mint 20 Mbyte. Ha tehát egy szállító 27 Mbyte-os winchester-t ígér, akkor a DOS alatt - formázva - 20 Mbyte tárolókapacitásunk lesz.

A fix lemezeket az első felhasználásuk előtt partícionálni és formázni kell (a DOS rendszerprogramok átmásolásával vagy anélkül).

A partícionálás során oszthatjuk fel a lemez területét a különböző operációs rendszerek között. Ha csak a DOS-t használjuk, egyetlen partíciót képzünk. Célszerű a számítógépünk első fix lemezére átvinni a DOS-t, mert ebben az esetben a rendszer betöltése floppy lemez behelyezése nélkül, a fix lemezről történhet.

A programok és az adatok a hajlékony és a fix lemezeken is file-okban helyezkednek el. Minden file-nak egyedi nevet kell adnunk; egy lemezen (pontosabban egy lemez adott fő- vagy alkönyvtárában) nem lehet két azonos nevű file.

A lemezeken található szektorok többségét a DOS a file-ok tárolására használja fel. A DOS alapprogramjai is (az egy szektoros betöltő rekordon kívül) file-okat képeznek, bár speciális jellegük következtében nem mindegyik jelenik meg a lemez tartalomjegyzékének kiiratásakor.

Honnan tudja a DOS, hogy egy lemezen milyen nevű file-ok találhatók, és ezek melyik lemezfelületen (melyik író-olvasó fej alatt), melyik sávokon, mely szektorokban helyezkednek el? Ezeket az adatokat a rendszer a lemez elkülönített részén kialakított tartalomjegyzékben (directory-ban) tartja nyilván. A tartalomjegyzék bejegyzései gyakran változhatnak, hiszen a file-okat módosíthatjuk, törölhetjük, átnevezhetjük, és új file-okat hozhatunk létre. A DOS mindig naprakész állapotban tartja a tartalomjegyzéket, így a felhasználónak nem kell az adatok fizikai elhelyezkedésével (fej-, sáv- és szektor-információkkal) foglalkoznia.

A nagykapacitású fixlemezek alkalmazásakor nem lenne célszerű az esetleg többszáz file-t ömlesztve egyetlen tartalomjegyzékbe bejegyezve tárolni (igen nehéz volna így a keresés, az eligazodás). Ezt a problémát küszöböli ki a hierarchikus (fastruktúrájú) tartalomjegyzékek használatának lehetősége. Ezzel a módszerrel elkülönítve kezelhetjük az egyes felhasználókhoz (vagy az egyes programokhoz, rendszerekhez) tartozó file-okat.

A tartalomjegyzéken kívül még egy speciális táblázatra van szüksége a DOS-nak, amely a lemez szektorainak (illetve a több szektorból álló, cluster-nek nevezett egységeknek) foglaltságáról tartalmaz információkat. Ez a file-helyfoglalási táblázat (angol neve: FAT, File Allocation Table). A FAT a lemeznek erre a célra fenntartott szektoraiban helyezkedik el, és a benne szereplő adatok alapján foglalja le (ill. szabadítja fel) a DOS a file-ok számára szükséges (ill.már nem igényelt) lemez-területeket.

2.3. A PC DOS indítása

A számítógépünk bekapcsolása után el kell indítanunk a DOS-t, mielőtt az operációs rendszert igénylő programokat futtatnánk. A rendszer betöltése hajlékony vagy fix lemezről történhet, a felhasználóra vár azonban a feladat, hogy szükség esetén a mindennapi használatra szánt rendszerlemezt kialakítsa. A rendszerlemez létrehozásáról és újraindításáról szól ez a rész.

2.3.1. A DOS elindítása

A DOS operációs rendszer elindításához szükségünk van a DOS rendszerlemezre (esetleg másolatára) vagy arra, hogy a számítógépben található első fix lemezegység aktív partícióján ott legyen a DOS. Arról, hogyan érhetjük el, hogy a DOS felkerüljön a fix lemezre, a következő részben szólunk.

Bekapcsolás után a gépbe beépített program először ellenőrzést végez, teszteli a gép fontosabb egységeit. Ha a teszt eredményes volt (nem jelenik meg hibaüzenet), akkor megkezdődik az operációs rendszer betöltése. A gép először az "A" jelű meghajtóval próbálkozik, de, ha a meghajtóban nincs lemez (vagy az egység ajtaja nyitva van), akkor az első (általában "C" jelű) fix lemezről kísérli meg betölteni a rendszert. Ha a hajlékonylemezes egységbe helyezett lemez nem tartalmaz önindítható programot, akkor a következő üzenet jelenik meg:

```
Non-System disk or disk error
Replace and strike any key when ready.
```

(Nem rendszerlemez vagy lemez-hiba.
Cserélje ki a lemezt, majd nyomja le bármelyik billentyűt.)

Ha ez a hiba lép föl, tegyük be a DOS rendszerlemezt az "A" meghajtóba, és nyomjunk le egy billentyűt. (Ha a winchester-en van a DOS, vegyük ki a lemezt az egységből és nyomjunk le egy billentyűt.

A DOS rendszerlemezen kívül egyéb automatikus indítású programok is vannak. Ezek, hasonlóan a DOS-hoz, a gép bekapcsolásával egyidejűleg betöltődnek a memóriába és elindulnak.

Ha az "A" meghajtó üres, és a géphez nem tartozik fix lemez (vagy a fix lemez nem tartalmaz operációs rendszert), de a gép ROM tárába be van égetve a BASIC, akkor a DOS helyett a BASIC interpreter lép működésbe.

Tudnunk kell, hogy a BASIC-nek ez a minimális változata (az ún. kazettás BASIC) nem képes sem a lemezes file-ok kezelésére, sem a kiterjesztett BASIC változat néhány fontosabb (pl. grafikus) funkciójának ellátására.

Az operációs rendszer sikeres betöltése után a DOS először a CONFIG.SYS file létezését ellenőrzi. Ha megtalálja, akkor a benne levő paraméterek alapján konfigurálja a rendszert, illetve betölti és inicializálja a benne definiált periféria-kezelő (device driver) programokat. Ezután vizsgálja meg, hogy az AUTOEXEC.BAT nevű file létezik-e a főtartalomjegyzékben.

Ha igen, akkor sorban végrehajtja az abban szereplő parancsokat. Tipikus esetben az AUTOEXEC futtatja le a rendelkezésre álló billentyűzetet kezelő segédprogramot (ha nem amerikai a billentyűzet), megkérdezi továbbá a dátumot és az időt. (A parancs-file-okról a 7. fejezet tartalmaz részletes leírást.)

Ha az indító lemezen nincs AUTOEXEC.BAT file, akkor a DOS csak a dátumot és az időpontot kérdezi meg a felhasználótól, erre célszerű mindig pontos választ adni. A dátumot a következőképpen kéri be a DOS:

```
Current date is Tue 1-01-1980
Enter new date:
```

Az aktuális dátumot hónap, nap, év sorrendben kell beírni (hacsak a dátum formátumát át nem állítottuk). Az évszázad, valamint a felesleges nullák elhagyhatók. 1986 június 12-én például így kell válaszolni:

```
6-12-86 <ENTER>
```

Az időpontot a DOS így kérdezi meg (a konkrét számok a példától természetesen eltérhetnek):

```
Current time is 00:00:02.13
Enter new time:
```

Az aktuális időt óra, perc másodperc, századmásodperc sorrendben lehet beírni, a gyakorlatban a másodpercet és a századmásodpercet elhagyhatjuk. Délután negyed háromkor így kell megadni a pontos időt:

14:15 < ENTER >

(A DATE és a TIME parancsokat részletesen a 6.4 és a 6.5 pontokban ismertetjük.)

Sok IBM PC XT-t és AT-t már belső órával (ami a gép működése közben töltődő akkumulátorral működik) szállítanak. Ebben az esetben az AUTOEXEC.BAT-be célszerű elhelyezni azt a programot, amelyik a belső óra idejét átírja az operációs rendszer órájába.

A dátum és az időpont beírása, illetve az AUTOEXEC file utolsó parancsának végrehajtása után a DOS megáll, és a parancsunkat várja. Ebben az állapotban a képernyő utolsó parancs-sorában csak a DOS prompt-ja (készenléti jele) látható:

A > (hajlékonylemezről) vagy
C > (fix lemezről)

A prompt jelzi, hogy a DOS a parancsainkat várja. A felszólító jel első betűje az ún. aktuális meghajtóra utal (erre a meghajtóra vonatkoznak majd a parancsok, ha nem szerepel bennük más meghajtó jele).

A prompt alakja - az AUTOEXEC.BAT file tartalmától függően - változhat, de a gyárilag szállított rendszerek általában nem tartalmaznak AUTOEXEC.BAT file-t, és így a DOS azonnal bejelentkezik. (Az AUTOEXEC.BAT file tetszőleges parancsokat is tartalmazhat. Ebben az esetben ezek a parancsok is végrehajtódnak, pl. elindulhat egy program, így előfordulhat, hogy a DOS prompt-ja egyáltalán nem jelenik meg.)

A DOS 3.00-ás és magasabb verziói a 10 Mbyte-nál nagyobb fix lemezegységeket a DOS 2.xx verzióitól eltérően kezelik. Ha például az "A" egységben levő DOS 2.11-es operációs rendszert elindítjuk és a "C" fix lemezegységet 3.00 alatt formáztuk meg, akkor üres tartalomjegyzéket kapunk! Ha nem tudjuk, hogy mi van egy lemezen, akkor mindig először onnan kíséreljük meg betölteni az operációs rendszert!

2.3.2 Ismételt indítás (restart)

Az előző pontban leírtak alapján bármikor el tudjuk indítani az operációs rendszert, ha a számítógép kikapcsolt állapotban volt. Előfordulhat azonban, hogy a gépet már bekapcsoltuk, és a DOS-t csak később akarjuk elindítani, vagy újraindítani. Néhány példa erre a szituációra:

- a gépen kazettás BASIC fut, és el akarjuk indítani a DOS-t,
- a jelenleg üzemelő DOS-tól eltérő verziószámú rendszert akarunk futtatni,
- módosítottuk a CONFIG.SYS leírást, és ezt aktivizálni szeretnénk,
- a DOS alatt futó program "lemerevedett", és más módszerrel nem tudunk kilépni belőle, ezért kell újraindítani a DOS-t.

A DOS újraindításához nem szükséges a számítógépet ki- és bekapcsolni. Jobban kíméli a gépet, és rövidebb ideig tart (kiesik a bekapcsoláskori ellenőrzés), ha az alábbi módszert választjuk:

Ha az újraindítást floppy-ról végezzük, helyezzük be a DOS rendszerlemezt az első meghajtóba. Fix lemez esetén erről történő indítás esetén nyissuk ki az első floppy-meghajtó ajtaját. Nyomjuk le ezután egyszerre a következő három billentyűt:

<Alt>, <Ctrl>, <Del>

A DOS újraindítása után megismétlődik a CONFIG.SYS és az AUTOEXEC.BAT file-ok feldolgozása is.

A fenti billentyűzetkombinációt szoftver úton érzékeli a rendszer, ezért ha a megfelelő megszakítási címeket valamelyik program átállította, a megnyomásnak semmilyen hatása sincs.

A gépek egy része tartalmaz <RESET> gombot. Ez nem a billentyűzeten, hanem magán a gépen található. Ennek megnyomása valóban egy „hardver reset" jelet generál. Ebben az esetben a gép teljes inicializálása megismétlődik.

2.3.3. Egyéb tennivalók

Az operációs rendszer eredeti (gyári) lemezeiről célszerű másolatot készíteni és azt használni. Egyes rendszerekhez szállítanak olyan parancs-file-t, ami ezt a feladatot elvégzi, de a legtöbb esetben magunknak kell ezt megtenni az alábbi módon:

1. Indítsuk el a DOS-t az eredeti lemezről (helyezzük a floppy-t az első meghajtóba, majd kapcsoljuk be a gépet)

2. Amikor megjelenik a DOS prompt-ja

```
A>
```

készítsünk másolatot a gyári lemezről. (Lehet, hogy a gép először a dátumot és az időpontot kérdezi meg. A kérdésekre most nem kell válaszolni, nyomjuk le az <ENTER> billentyűt.) Ha két meghajtónk van, tegyünk a másodikba egy üres lemezt (lehet formázatlan is). Ha csak egy meghajtóval rendelkezünk, akkor az üres lemezt helyezzük készenlétbe. Írjuk be ezután a következő parancsot:

```
DISKCOPY A: A:<ENTER>
```

illetve két meghajtó esetén:

```
DISKCOPY A: B:<ENTER>
```

(Az <ENTER> jelölés arra utal, hogy le kell nyomni az <ENTER> billentyűt. Vigyázzunk, mert a billentyűk felirata nem szükségszerűen egyezik meg a képernyőn megjelenő betűkkel, pl. az Y és a Z betűket esetleg fordítva kell használnunk.) Egymeghajtós gépeknél a gyári és az új lemezt esetleg többször is cserélni kell a meghajtóban, erre a DOS fel is szólít. Végül megjelenik az alábbi kérdés:

```
Copy another (Y/N)?
```

Nyomjuk le válaszként az <N> billentyűt. (A DISKCOPY parancs részletes leírása a 4.12 pontban található.)

3. Tegyük az eredeti DOS lemezt biztonságos helyre és a további munkát a másolattal folytassuk.

4. Készítsük el a DOS másolaton az önindító (AUTOEXEC.BAT) file-t. Az alábbi példában feltételezzük, hogy német billentyűzet tartozik a géphez. Ha amerikai billentyűzetünk van, egyszerűen hagyjuk ki a KEYBGR szót az eljárásból. Ha a billentyűzet más típusú, akkor a KEYBGR helyett a megfelelő segédprogram nevét kell leírni. Helyezzük az első meghajtóba a DOS másolatot, és írjuk be a következőket:

```
COPY CON AUTOEXEC.BAT <ENTER>
KEYBGR <ENTER>
DATE <ENTER>
TIME <F6> <ENTER>
```

(Az <F6> jelölés arra utal, hogy le kell nyomni az <F6> billentyűt.) Ragasszunk ezután írásvédő csíkot a lemezre, és ezentúl használhatjuk a DOS indítására.

2.3.4 Fix-lemezegység használata

Ha a gépünk fixlemez-meghajtóval is fel van szerelve, akkor célszerű az első (általában "C" betűjelű) fix lemezből rendszerlemezt készíteni, hogy a DOS-t a hajlékonylemezes egység használata nélkül is el lehessen indítani.

A fix lemezeket használatbavételük előtt partícionálni és formázni kell. A partícionálást (a lemez területének felosztását a különböző operációs rendszerek között) az FDISK paranccsal lehet elvégezni. Az alábbi példában a lemez teljes kapacitását a DOS alá rendeljük, mert - egyelőre - csak a DOS operációs rendszert akarjuk használni. Tegyük be az előbbiek során készült DOS másolatot az első floppy-meghajtóba, majd írjuk be:

```
FDISK <ENTER>
```

A képernyőn megjelenő menü első pontját választjuk (DOS partíció létrehozása):

```
1 <ENTER>
```

Az ezután megjelenő újabb menüből ismét az (1.) pontot válasszuk ki. (Elsőrendű DOS partíció létrehozása.) *A 3.30-asnál korábbi verzió esetén ez a kérdés kimarad.*

A következő kérdésre (a teljes lemezt a DOS-szal akarja használni?) igenlő választ adunk:

```
Y <ENTER>
```

A képernyőn a következő üzenet jelenik meg:

```
System will now restart.
Insert DOS diskette in drive A:
Press any key when ready...
```

Tetszőleges billentyű megnyomására a rendszer újraindul. Ha az első (C jelű) lemezegységet particionáltuk, akkor a hajlékonylemezes egységbe be kell helyeznünk a DOS rendszerlemezt, s csak ekkor nyomjunk meg egy billentyűt. Ezután a program újraindítja a DOS-t, amely - a sikeres partícionálás után - már kezelni tudja a fix lemezt.

(Az FDISK parancs részletes ismertetése az 5.1 pontban olvasható. A fix lemezek további partícionálását is az 5.1 pont alapján kell elvégezni.)

Ezután elkezdhetjük a fix lemez formázását. Az első (példánkban "C" jelű) lemezre átmásoljuk a DOS-t is. A többi fix lemez esetében erre nincs szükség. Írjuk be a következőket:

```
FORMAT C:/S/V <ENTER>
```

Fix lemez formázásakor a rendszer figyelmeztető üzenetet küld, majd visszakérdez, hogy biztosan ezt akarjuk? Az "Y" billentyű megnyomására megkezdődik a formázás, amelynek előrehaladását a képernyőn is nyomon tudjuk követni. (Csak 3.20-as vagy későbbi verzió esetén.) Amikor a formázással végezve a gép a fix lemeznek adandó nevet (volume label) kérdezi, írjuk be egy maximum tizenegy betűs nevet (pl. lacilemeze), majd nyomjuk le az <ENTER>-t.

A fix lemez megformázása a lemez (pontosabban a DOS partíció) jellemző adatainak kiírásával ér véget:

```
21309440 bytes lokal disk space
   79872 used by system            (függ a használt DOS verziójától!)
21229558 bytes available on disk
```

A rendszer által elfoglalt hely függ a DOS használt verziójától. Például DOS 3.00 esetén csak 61440 byte.

Végül másoljuk át a DOS-t tartalmazó lemezről a DOS segédprogramjait:

```
COPY *.* C: <ENTER>
```

Ezután - a gép legközelebbi bekapcsolásakor - a 2.3.1-ben leírt eljárással már a fix lemezről is indíthatjuk a DOS-t.

A fenti eljárás valamennyi DOS file-t a C: lemezegység főtartalomjegyzékében helyezi el. Ennél célszerűbb a következő parancsok kiadásával egy DOS nevű alkönyvtárat létrehozni, és abban elhelyezni a DOS file-jait:

```
COPY COMMAND.COM C: <ENTER>
MD C:\DOS <ENTER>
CD C:\DOS <ENTER>
COPY A:*.* C:\DOS <ENTER>
```

Ebben az esetben azonban az AUTOEXEC.BAT file-ban el kell helyezni a

```
PATH C:\DOS
```

parancsot, hogy az operációs rendszer mindig megtalálja a DOS külső parancsait. Az AUTOEXEC.BAT file csak akkor működik rendeltetés-szerűen, ha a főtartalomjegyzékben helyezzük el!

2.4 A billentyűzet kezelése, DOS parancsok kiadása

Mielőtt dolgozni kezdenénk a DOS-szal, meg kell ismerkednünk a billentyűzet használatával, az egyes billentyűk funkciójával. Ebben a pontban csak a DOS billentyűzet-kezelésével foglalkozunk, és felhívjuk arra a figyelmet, hogy más programok (pl. a BASIC interpreter, vagy a különböző szövegszerkesztők) a leírtaktól teljesen eltérő funkciót rendelhetnek az egyes billentyűkhöz. Ezért az alábbiakban ismertetett módszereket csak a DOS parancsok kiadásakor és a DOS segédprogramok, köztük a sorszerkesztő (EDLIN) és a hibakereső program (DEBUG) futtatásakor alkalmazhatjuk változtatás nélkül.

A PC DOS-t futtató számítógépekhez **különböző billentyűzeteket forgalmaznak.** *Nálunk a leggyakoribb az amerikai és a német elrendezésű billentyűzetek használata.* Ezek nemcsak a jelek elhelyezésében térnek el egymástól, hanem a következő szempontokból is különböznek:

- A német billentyűzeten bizonyos speciális jelek (pl. kapcsos zárójelek) helyett ékezetes betűk (ö, ü, ä) találhatók.
- A német billentyűzeten - az amerikaival ellentétben - egyszerűen előállíthatók egyes ékezetes betűk (köztük a magyar á, é, í, ó, ú, és É).
  Ezek képernyős megjelenítéséhez először le kell nyomnunk az ékezet-billentyűt, majd azt a gombot, amelyiken az alapjel szerepel. Így például az ó betűt az ' (aposztrof) és az o billentyűk egymásutáni lenyomásával állíthatjuk elő.
- A német billentyűzeten két speciális jel (a @ és a \) a billentyűk oldalára van írva. Ezeket úgy lehet megjeleníteni, hogy az <ALT> és a <CTRL> billentyűk lenyomva tartása közben megnyomjuk az adott jelet ábrázoló billentyűt.
- Az Y és Z betűk fordítva - a magyar írógép-szabványnak megfelelően - vannak.

A billentyűzet nagyrésze az írógépről ismerős betűket, számokat és jeleket tartalmazza. A szóközbillentyű (space) is a megszokott helyén van. Vigyázni kell azonban, nehogy a felső sorban megtalálható 0 es 1 számokat az o, illetve l betűkkel helyettesítsük.

A nagybetűk, valamint a billentyűkön felülre írt jelek előállításához a <Shift> billentyűt kell nyomva tartani a megfelelő gomb lenyomása alatt. (A <Shift> az írógép nagybetűváltójának felel meg, a billentyűzet bal és jobb oldalán található. Jele: Shift vagy egy felfelé mutató nyíl.) A DOS parancsok kiadásakor egyébként mindegy, hogy nagy- vagy kisbetűket használunk, sőt a DOS még a vegyes írásmódot is elfogadja. Amikor viszont szövegeket, forrásprogramokat állítunk elő pl. az EDLIN programmal, akkor már értelme lehet a kis- és nagybetűk megkülönböztetésének.

Ha tartósan csak nagybetűket akarunk használni, nyomjuk le a <Caps Lock> (nagybetűzár) billentyűt. Ezután - egészen e billentyű újbóli megnyomásáig - valamennyi betű nagybetűs formában jelenik meg. (Ez csak a betűkre vonatkozik; a billentyűkön felülre írt speciális jelek előállításához továbbra is le kell nyomni a <Shift>-et!) Ha a <Caps Lock> lenyomása után kisbetűt akarunk írni, a <Shift> billentyűt kell használnunk.

A betűktől jobbra helyezkedik el az igen fontos <ENTER> billentyű. (Egyes gépeken a RETURN nevet viseli.) Minden DOS parancs beírása után le kell nyomnunk az <ENTER>-t, innen tudja a DOS, hogy vége a parancs beírásának, és a végrehajtást csak ezután kezdi meg.

Az < ENTER > -től jobbra található numerikus billentyűknek (0-tól 9-ig, valamint a tizedespont) kettős szerepe van. Alaphelyzetben nem a számok megjelenítésére szolgálnak, hanem a feliratuk alsó sorában szereplő funkciókat hajtják végre. Átkapcsolásuk számírásra, majd visszakapcsolásuk a < Num Lock > (számzár) billentyűvel történhet. DOS alatt e billentyűket általában az alaphelyzetükben használjuk.

A numerikus billentyűk mellett látható + és - billentyűk csak megismétlik a klaviatúra középső részén is megtalálható azonos jeleket.

Ennyi információ birtokában már bátran kiadhatjuk az első DOS parancsot. Tegyük fel, hogy az AUTOEXEC.BAT nevű file-t akarjuk az egyik floppy-ról a másikra átmásolni. Erre a célra a COPY parancs szolgál, amelyet részletesen a 3. fejezetben ismertetünk. Ezt kellene tehát beírnunk:

```
COPY A:AUTOEXEC.BAT B: < ENTER >
```

Ha a fenti parancsot hibátlanul írtuk le, biztosan sikerül a másolás. Mit tehetünk azonban, ha - még az < ENTER > lenyomása előtt - észrevesszük, hogy gépelési hibát vétettünk? A legegyszerűbb javítási mód, ha a < Backspace > (visszalépés, visszatörlés) billentyűvel töröljük a parancs hibás részét, majd beírjuk a helyes szöveget. **(A < Backspace > billentyű a legfelső sorban helyezkedik el, jele általában egy balra mutató nyíl.** Minden lenyomásakor egy-egy jelet töröl visszafelé, azaz balra.) Ha például az előző parancsot véletlenül így írtuk (kihagyva az E betűt):

```
COPY A:AUTOXE
```

a javításhoz nyomjuk le kétszer a < Backspace > -t (az utolsó két hibás betű törlésére):

```
COPY A:AUTO
```

majd folytassuk a parancsot:

```
COPY A:AUTOEXEC.BAT B: < ENTER >
```

(Megjegyezzük, hogy a numerikus billentyűzeten a balra mutató nyilat - és a 4-es számot - ábrázoló billentyű is a < Backspace > -szel azonos szerepet tölt be.)

Ha a hibás parancs után már lenyomtuk az < ENTER > -t, akkor általában maga a DOS veszi észre a hibát (ha értelmetlen utasítást kapott), ilyenkor hibajelzést ad. Ha viszont valahogyan értelmezni tudja a beírt szöveget, akkor elkezdi a végrehajtást.

Az éppen futó programot a következőképpen lehet félbeszakítani (ha tévedésből adtuk ki a parancsot, vagy már nincs szükség a program további futására): tartsuk lenyomva a < Ctrl > billentyűt, majd nyomjuk meg a < Scroll Lock > gombot (oldalán többnyire látható a Break, megszakítás felirat). Ez a legtöbb esetben félbeszakítja az adott program futását. (Ugyanez a hatása a < Ctrl > -C kombinációnak is.)

Tegyük fel, hogy beírtunk egy DOS parancsot, de még nem "küldtük el" az < ENTER > megnyomásával. Rájövünk azonban, hogy a parancs végrehajtására most nincs szükség, vagy látjuk, hogy a parancs annyi hibát tartalmaz, hogy még korrigálni sem érdemes. Hogyan törölhetjük ki az egész sort? Az egyik lehetséges, már ismert megoldás, ha annyiszor lenyomjuk a < Backspace > billentyűt, ahány betűt (karaktert) a parancs tartalmazott. A gyorsabb módszer: az < Esc > feliratú billentyű lenyomása; ez ugyanis az egész sort törli.

A hibás sor ugyan a képernyőn marad (egy \-rel, blackslash-sel lezárva), de végrehajtására nem kerül sor, hanem a gép új sorba lép, és következő parancsunk beírására vár. (Az Esc felirat az angol escape = menekülés, kilépés szó rövidítése.)

Léteznek az eddig leírtaknál bonyolultabb, de hatékonyabb szerkesztési műveletek is, amelyek kihasználják, hogy a DOS "emlékezik" az utoljára beírt parancs-sorra, és így annak szövegét a következő parancs kiadásakor - részben vagy egészben - újra felhasználhatjuk. Ezeket a módszereket a DOS parancsok kiadásánál is felhasználhatjuk, de fontosabb szerepet a sorszerkesztő (EDLIN) program futásánál játszanak. Ezért a funkciós billentyűk, az < Ins > és a < Del > gombok ismertetését a fejezet első olvasásakor bátran átlapozhatjuk. Az EDLIN használata előtt viszont érdemes e billentyűkkel is megismerkedni.

A klaviatúra bal oldalán (egyes billentyűzeteken ettől eltérő helyen) található < F1 >, < F2 >,..., < F10 > feliratú gombokat **funkciós billentyűknek nevezzük** (az egyes programok ugyanis különböző funkciókat rendelhetnek hozzájuk). **A DOS-ban csak az első hat funkciós billentyű használható.**

Az **< F1 > billentyű lenyomásának hatására az előző DOS parancs** (vagy EDLIN sor) **egy-egy betűje** (karaktere) **ismétlődik meg**. Ha pl. két file-másolást végzünk egymás után, és a DOS a

```
COPY A:AUTOEXEC.BAT B:
```

parancsot már végre is hajtotta, akkor a következő másoló parancs beírását az < F1 > billentyű megkönnyíti.

Tegyük föl, hogy most a COMMAND.COM nevű file-t akarjuk átmásolni az A meghajtóról a B-re. Az előző parancsból így változtatás nélkül átvehetjük az első hét betűt (COPY A:), ehhez nyomjuk le hétszer az < F1 > gombot. A képernyőn ez jelenik meg:

```
COPY A:
```

A parancs többi részét a szokásos módon kell beírnunk:

```
COPY A:COMMAND.COM B:< ENTER >
```

(Megjegyezzük, hogy a numerikus billentyűzeten a jobbra mutató nyilat és a 6-os számot ábrázoló billentyű is az < F1 > -gyel azonos szerepet tölt be.)

A fenti feladatra az < F2 > billentyűt is felhasználhattuk volna. Az < F2 > ugyanis szintén az előző DOS parancs karaktereit ismétli meg, de nem egyesével, hanem az általunk megadott karakterig. **Az < F2 > leütése után mindig meg kell nyomni annak a karakternek a gombját is, amelyiknek az első előfordulásáig a másolást kérjük** (ez a karakter már nem vesz részt a másolásban).

A COMMAND.COM másolására vonatkozó parancsot az < F2 > -vel a következőképpen adhatnánk meg (feltéve, ha az előző parancs az AUTOEXEC.BAT file másolása volt). Az < F2 > lenyomása után meg kell adnunk azt a karaktert, ahol a másolást be kívánjuk fejezni. A megadott karakter már nem vesz részt a másolásban, ezért az AUTOEXEC szó első A betűjét jelölhetnénk ki (ez az első betű, amelyre már nincs szükségünk). A másolás viszont mindig a kijelölt karakter első előfordulásáig tart, ezért az A betű lenyomása esetén csak a COPY karaktersorozat másolódna át. Érdemesebb ezért az < F2 > után a ":" (kettőspont) billentyűt megnyomni, mert ekkor ez jelenik meg a képernyőn:

```
COPY A
```

A parancs további részét a szokásos módon be kell írnunk:

```
COPY A:COMMAND.COM B:<ENTER>
```

Az < F3 > billentyű teljes egészében megismétli az előző DOS parancsot, feltéve, hogy a billentyű lenyomásakor a sor első pozíciójánál tartunk. Ha a parancs-sorban már tovább léptünk, akkor csak az adott pozíciótól kezdve másolja át az előző parancs karaktereit. Az < F3 > tehát igen kényelmes eszköz a parancsok változatlan megismétlésére is.

Az < F4 > billentyű az < F2 > ellentétének tekinthető. **Az < F4 > törli az előző parancsnak a megadott karakter első előfordulásáig terjedő részét** (a megadott karaktert már nem törli). Ezután az előző parancsból megmaradt részt az < F1 >, < F2 > vagy < F3 > billentyűk segítségével másolhatjuk át az új parancs-sorba.

Nézzünk meg egy összetett példát, amelyben az < F4 > billentyű is szerepel.

Tegyük fel, hogy az előző DOS parancs ez volt:

```
COPY A:AUTOEXEC.BAT B:
```

Ha ezek után az AUTO.BAT nevű file-t akarjuk átmásolni, nyomjuk le először az < F2 >, majd az E billentyűket. Erre átmásoljuk az előző parancsból az első E betűig terjedő részt:

```
COPY A:AUTO
```

Most jön a törlés: ki akarjuk hagyni az EXEC szót. Nyomjuk meg az < F4 >-et, majd a "." (pont) billentyűt. Ez törli az első parancsból a pontig terjedő részt, de magát a pontot nem. A képernyőn a törlést nem látjuk. Végül át kell másolnunk az előző parancs hátralevő részét (.BAT B:). Ezt a műveletet az < F3 > lenyomásával végezzük el. A parancs-sorban már a kész parancsot látjuk:

```
COPY A:AUTO.BAT B:
```

Ha most lenyomjuk az < ENTER >-t, a DOS végrehajtja a parancsot.

Az < F5 > billentyű lenyomásával azt érjük el, hogy az általunk beírt, de még el nem küldött parancs-sor helyettesíti a DOS memóriájában (input pufferében) az előzőleg beírt parancsot. Az új parancs végrehajtására ekkor még nem kerül sor, az eredmény csak annyi, hogy az < F1 >, < F2 >, < F3 >, < F4 >, < Ins >, < Del > billentyűkkel ezek után az új parancs-sort lehet tovább feldolgozni, átszerkeszteni. Az < F5 > használatára az < Ins > és a < Del > billentyűk ismertetése után mutatunk példát.

Az < Ins > (insert, beszúrás) és a < Del > (delete, törlés) billentyűk segítségével új karaktereket illeszthetünk be, illetve meglévő karaktereket törölhetünk ki az átszerkesztés alatt álló sorban.

Az < Ins > billentyű lenyomása után a gép "beszúró üzemmódba" tér át, és ettől kezdve a beírt karaktereket beilleszti a sor adott helyére, miközben a tőle jobbra álló karaktereket mindig jobbra tolja. A beszúró módból az <Ins> ismételt lenyomásával vagy bármely más szerkesztő billentyűvel tudunk kilépni.

A <Del> billentyű minden lenyomása egy-egy karaktert töröl az előző (átszerkesztés alatt álló) **sorból.** Eközben a törölt karaktertől jobbra elhelyezkedő szöveg mindig eggyel balra tolódik.

Az alábbi példában az <F5>, az <Ins> és a <Del> szerepét is bemutatjuk. Tegyük föl, hogy ismét a

```
COPY A:AUTOEXEC.BAT B:
```

parancsot szeretnénk kiadni. Gépelési hibák következtében azonban a

```
COOPY A;AUTOEXEC.BT B:
```

szöveg áll a parancs-sorban (egy O felesleges, kettőspont helyett pontosvessző van, és egy A hiányzik). Nem lenne érdemes a <Backspace>-szel visszatörölni az első O betűig, sem pedig az <Esc> gombbal törölni az egész sort, mert akkor mindent újra kell gépelni. Ehelyett lenyomjuk az <F5> billentyűt, amelynek hatására a hibás sor végére a DOS a @ jelet írja, és egy új, üres sorba tér át, ahol elkezdhetjük a hibás sor átszerkesztését. Az első két betű hibátlan, ezeket az <F1> kétszeri lenyomásával ismételjük meg:

```
CO
```

A következő betű, a második O felesleges, ezért a <Del> billentyű egyszeri lenyomásával töröljük. Ez nem látszik a képernyőn. Ismét az <F1>-et kell használnunk a következő négy hibátlan karakter átmásolására (négyszer lenyomjuk az <F1>-et):

```
COPY A
```

A pontosvesszőt nem akarjuk átvenni, helyette lenyomjuk a kettőspont billentyűt. Ez felülírja, helyettesíti az eredeti parancs azonos helyén álló karakterét:

```
COPY A:
```

Az AUTOEXEC.B szöveget az <F1> tízszeri lenyomásával másoljuk át:

```
COPY A:AUTOEXEC.B
```

A hiányzó A betűt a B és a T közé akarjuk beilleszteni. Ehhez nyomjuk le az <Ins> gombot, majd írjuk le az A betűt. Ezután lépjünk ki a beszúró módból az <Ins> újbóli megnyomásával:

```
COPY A:AUTOEXEC.BA
```

A parancs hátralevő részét változtatás nélkül akarjuk megismételni, ez az <F3> funkciója, nyomjuk le:

```
COPY A:AUTOEXEC.BAT B:
```

Most már hibátlan a parancs, le lehet nyomni az <ENTER>-t.

Ezzel befejeztük a szerkesztő billentyűk tárgyalását, és áttérünk az egyéb feladatokat ellátó speciális billentyűk ismertetésére.

A <PrtSc> (Print Screen, képernyő nyomtatása) felirat általában a csillag-jel (*) fölött szerepel. Első funkciója ezért a <Shift>-tel együtt aktivizálható. **A <Shift-PrtSc> kombináció lenyomásakor a képernyő tartalma kinyomtatódik a számítógép első printer-adapterére kapcsolt nyomtatón** (amennyiben ez be van kapcsolva). Ha a nyomtató nincs bekapcsolva, hibaüzenetet nem kapunk.

A <Ctrl> és a <PrtSc> együttes lenyomásának más a hatása. Ezzel a kombinációval indíthatjuk el az ún. echo (visszhang) üzemmódot. Echo-módban a printeren minden sor megjelenik, amit a billentyűzeten beírunk, valamint az is, amit a számítógép ír a képernyőre. Az echo-módból a <Ctrl-PrtSc> ismételt megnyomásával léphetünk ki. (Ugyanez a szerepe a <Ctrl-P> kombinációnak is.)

Vigyázzunk a <Ctrl-PrtSc> használatakor, hogy a nyomtató be legyen kapcsolva. Ha nincs bekapcsolva - vagy egyéb hiba lép fel -, a DOS hibaüzenetet ír ki (a szokásos abort/retry/ignore? kérdéssel). Ilyenkor hiába akarjuk a parancsot megszüntetni (abort válasz), mivel a nyomtatás echo be van kapcsolva, újabb hibaüzenetet kapunk - ui. a hibaüzenetet is ki akarja nyomtatni a DOS. Ekkor két dolgot tehetünk: vagy bekapcsoljuk a nyomtatót, vagy ismételten (egyszer!) lenyomjuk a <Ctrl-PrtSc> billentyűket, majd addig válaszolunk i-t (ignore) a hibaüzenetekre, míg azok meg nem szűnnek. A <Ctrl-PrtSc> billentyű grafikus üzemmódban csak akkor működik helyesen, ha előtte a GRAPHICS DOS parancsot kiadtuk.

A <Shift-PrtSc>-t a BIOS – a billentyűzet-megszakítást figyelő rutin –, a <Ctrl-PrtSc>-t a DOS kezeli le. Vigyázzunk, hogy - hacsak nem használunk speciális programot erre - grafikus képernyő-kép esetén ne használjuk a <Shift-PrtSc>-t!

Ha a számítógép túl sok információt ír ki egyszerre, és a felső sorok eltűnnek a képernyőről, mielőtt elolvashatnánk azokat, nyomjuk meg egyszerre a <Ctrl-NumLock> billentyűket. Ez megállítja a képernyőre (és a printerre) irányuló listázást, és időt ad a kiírt adatok megtekintésére. A program működése ezzel nem szakad véglegesen félbe (mint a <Ctrl-Break> használatakor), hanem bármelyik betű vagy számbillentyű lenyomásakor ott folytatódik, ahol abbamaradt. A <Ctrl-NumLock> helyett a <Ctrl-S> billentyűt is megnyomhatjuk, a hatás ugyanaz.

Az <F6> billentyűvel lehet file-vége jelet előállítani. A szövegszerkesztők (editorok) és más programok általában kiteszik a file végére e jelet, de egyes esetekben erről magunknak kell gondoskodnunk (például a COPY parancs használatakor, ha az input a billentyűzetről érkezik). Az <F6>-éval megegyezik a <Ctrl-Z> billentyűk hatása is.

Kevésbé fontos a <Ctrl-ENTER> kombináció, amelynek lenyomásakor az írást a következő sorban folytathatjuk anélkül, hogy az előző sort a DOS végrehajtaná.

Az alábbi táblázat röviden összefoglalja a speciális billentyűk és billentyűkombinációk szerepét a DOS-ban:

Billentyű	**Hatása**
< Alt-Ctrl-Del >	DOS újraindítása
< Ctrl-Break > vagy < Ctrl-C >	Futó program, parancs megszakítása
< Ctrl-NumLock > vagy < Ctrl-S >	Listázás félbeszakítása
< Backspace >	Utolsó karakter törlése (balra)
< ENTER >	DOS parancs (sor) vége
< Ctrl-ENTER >	Új sorba lép át
< Esc >	Sor törlése (az előző sor nem változik)
< F1 >	Egy karakter másolása az előző sorból
< F2 >	Másolás megadott karakterig
< F3 >	Előző sor ismétlése
< F4 >	Törlés megadott karakterig
< F5 >	Az előző sort az új sorral helyettesíti
< F6 > vagy < Ctrl-Z >	File-vége jel
< Ins >	Beszúró mód be/kikapcsolása
< Del >	Egy karakter törlése
< Shift-PrtSc >	A képernyő tartalmának nyomtatása
< Ctrl-PrtSc > vagy < Ctrl-P >	Nyomtatóra írás (echo) be/kikapcsolása

2.5 Lemez- és file kezelés

2.5.1 A fastruktúrájú tartalomjegyzék

A lemez kapacitásának növekedése azt eredményezte, hogy a lemezen egyre több és több file-t lehetett elhelyezni. A nagyméretű tartalomjegyzék (directory) mind a rendszernek, mind a felhasználónak kényelmetlen. Ezért kialakították a többszintű, ún. fastruktúrájú tartalomjegyzéket, amit néha „könyvtár"-nak is neveznek.

A lemez formázása után a rendszer a lemezen előállít egy tartalomjegyzéket. Ezt szokták rendszer-, vagy főtartalomjegyzéknek, vagy gyökér tartalomjegyzéknek (root directory) is nevezni. Amikor a DOS-t elindítjuk a számítógépen, a gyökér tartalomjegyzékben vagyunk.

A tartalomjegyzék mérete:

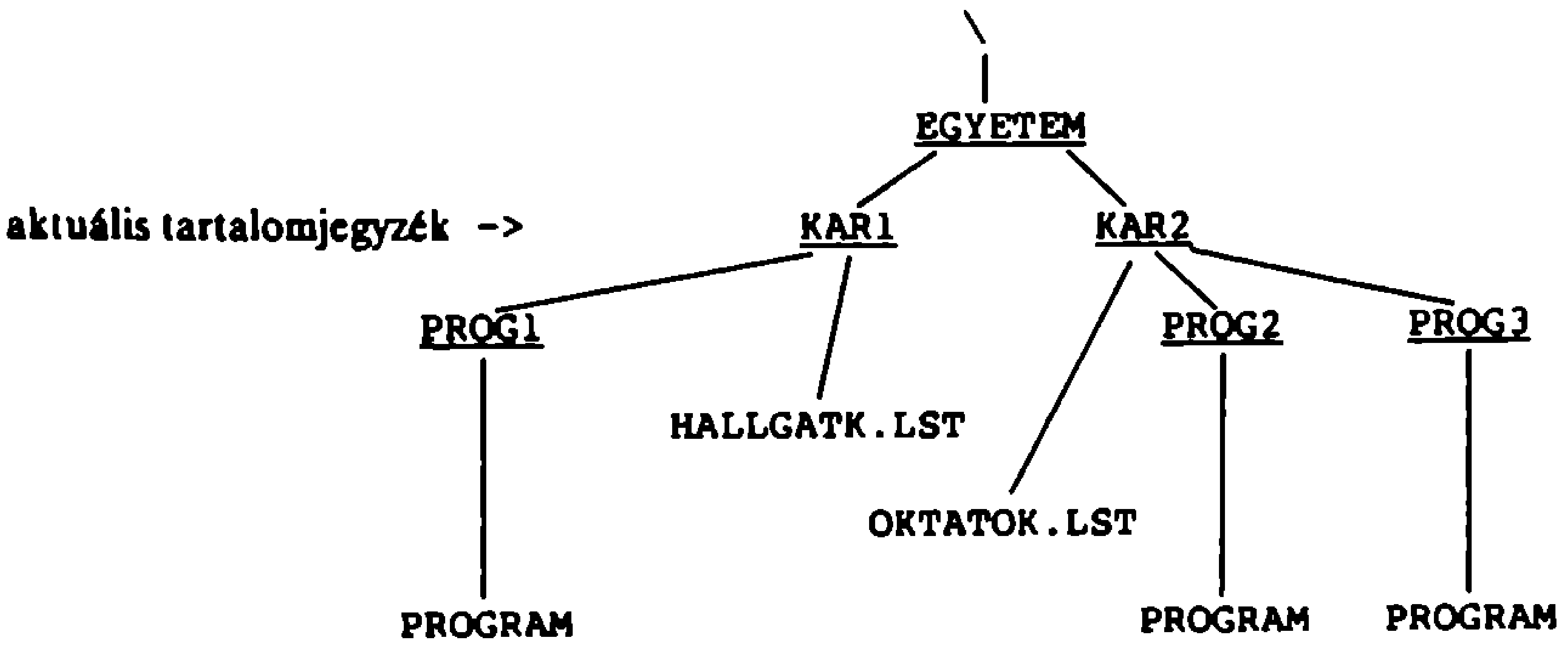
- 64 db bejegyzés 1 oldalú floppy,
- 112 db bejegyzés 2 oldalú floppy,
- 224 db bejegyzés magas kapacitású (high density) floppy esetén,
- fix lemez esetén ez a DOS partíció méretétől függ, de legfeljebb 512 bejegyzés lehet a főtartalomjegyzékben.

A tartalomjegyzékbe nemcsak file-nevek kerülhetnek, hanem további tartalomjegyzék-nevek is. Ezek a tartalomjegyzékek szintén tartalmazhatnak további tartalomjegyzék-azonosítókat.

Ez utóbbiakat altartalomjegyzéknek nevezzük. Az altartalomjegyzékek mérete nem korlátozott, tetszőleges számú file-név és további altartalomjegyzéknév bejegyzése lehetséges. Méretét csak a rendelkezésre álló lemezterület korlátozza.

A többszintű tartalomjegyzék-rendszert floppy lemezen is ki lehet alakítani, de gyakorlati jelentősége főleg fix lemez esetén van. Normál esetben minden felhasználónak lehet egy vagy több könyvtára (tartalomjegyzéke). Mindenki csak a saját, és a megállapodások szerinti közös tartalomjegyzéket használhatja. Az előny nyilvánvaló.

Egy lehetséges tartalomjegyzék-struktúra a következő:



Jelölés: Az aláhúzott azonosítók altartalomjegyzéket, az aláhúzás nélküliek pedig fileokat jelölnek.

A példában a lemezen több azonos file-név is van. Ez azért lehetséges, mert különböző altartalomjegyzékben találhatók.

A következőkben néhány alapfogalmat ismertetünk a fastrukturájú tartalomjegyzékekkel kapcsolatban.

Aktuális tartalomjegyzék

A rendszernek a következő két információra mindig szüksége van:

- melyik az aktuális meghajtó,
- melyik az aktuális tartalomjegyzék.

A rendszer indítása után az aktuális meghajtón levő lemez gyökér tartalomjegyzékét tekinti aktuális tartalomjegyzéknek. Az aktuális tartalomjegyzék paranccsal megváltoztatható (CHDIR/CD parancs).

File-ok elérése

A rendszernek a file-ok kezeléséhez a következő információkra van szüksége:

- a meghajtó azonosítója,
- tartalomjegyzék megadása, amiben a file található,
- a file azonosítója.

Ha a file az aktuális tartalomjegyzékben található, akkor a tartalomjegyzék azonosítóját nem kell megadni. Ha nem az aktuális tartalomjegyzékben található, akkor meg kell adni a kívánt tartalomjegyzékhez vezető utat, amelyet a tartalomjegyzék-nevek segítségével írunk le.

Az út megadását a fő-, az aktuális- vagy az aktuálist követő tartalomjegyzékkel kezdjük. Az út tartalomjegyzék-nevek sorozatából áll, amelyet \ (backslash) választ el egymástól. Az út a következő formában adható tehát meg:

[\][*altartalomjegyzék-név*\]...

Az út elején álló \ jel azt jelenti, hogy az út nem az aktuális tartalomjegyzéktől, hanem a gyökértől indul. Az út megadásánál két speciális tartalomjegyzék-nevet használhatunk. Az egyik a . (egyetlen pont), ami az aktuális tartalomjegyzéket jelenti. A másik a .. (dupla pont), ami az aktuális tartalomjegyzéket tartalmazó tartalomjegyzék.

(A fenti példában az aktuális tartalomjegyzék a KAR1. A .. tartalomjegyzék így az EGYETEM.)

A file-ok megadása a fentiek alapján a következőképpen történik:

[*d*:][*út*]*file-név*[*.kit*]

A *d*: a lemezegység egybetűs azonosítója. Lehetséges értéke a rendszerben levő lemezegységek számától és típusától függ (rendszerint C, D vagy E).

A fenti struktúra esetén a bal szélső PROGRAM file-ra az alábbi módokon hivatkozhatunk:

PROG1\PROGRAM (az aktuális KAR1 tartalomjegyzékből)

vagy

\EGYETEM\KAR1\PROG1\PROGRAM (bárhonnan)

A jobb szélső PROGRAM file az alábbi módon érhető el:

..\KAR2\PROG3\PROGRAM (az aktuális KAR1 tartalomjegyzékből)

vagy

\EGYETEM\KAR2\PROG3\PROGRAM (bárhonnan)

2.5.2 File- és tartalomjegyzék nevek

A file-ok azonosítója két részből áll:

- file-név,
- file-név kiterjesztés (kiterjesztés).

A file-név maximálisan nyolc karakter lehet és a következő jelekből állhat:

- az angol ABC betűi (a-z, A-Z),
- számok: 0-9 és
- a következő speciális jelek: $ # & @ ! % ( ) ' - _

A file-név kiterjesztése mindig a file-név mögött áll, és mindig ponttal (.) kezdődik. A hossza a ponttal együtt maximálisan négy karakter. A kiterjesztés ugyanazokból a jelekből állhat, mint a file-név. A file-név és a hozzá tartozó kiterjesztés együtt alkotja a **file-azonosítót**.

A tartalomjegyzék-név ugyanilyen szabályoknak kell, hogy eleget tegyen. Így a tartalomjegyzék neve is tartalmazhat kiterjesztést, bár ezzel a lehetőséggel ritkán szoktunk élni.

A kiterjesztés a file-azonosítónak nem kötelezően alkalmazandó része. Ha azonban a file létrehozásakor megadtuk, mindig célszerű használnunk. A BASIC például, ha csak filenevet adunk meg, és az hosszabb, mint 8 karakter, a maradékból hozza létre a kiterjesztést.

Például a SZOVEG.TXT és a SZOVEG két különböző file-t azonosít (és mindkettő megengedett azonosító).

Néhány rendszerprogram, fordítóprogram és egyes alkalmazói rendszerek egy adott feladatra meghatározott kiterjesztést használnak. Ezeket felhasználói file-ok azonosításánál nem célszerű felhasználni. A leggyakrabban használt kiterjesztéseket a D. függelékben soroljuk fel.

Ha nem emlékszünk pontosan, vagy nem akarjuk pontosan megadni a file-azonosítót, két speciális karaktert, a csillagot (*) és a kérdőjelet (?) is alkalmazhatjuk a file nevében vagy kiterjesztésében.

A kérdőjel azt jelöli az azonosítóban, hogy helyén bármilyen más megengedett jel állhat. Így minden olyan file kiválasztásra kerül, amelyeknek azonosítója a kérdőjelek helyét kivéve megegyezik.

A csillag (*) a file-azonosítójában azt jelenti, hogy ezen a helyen, valamint a file-név és/vagy a file-kiterjesztés ezt követő pozícióján vagy pozícióin bármilyen megengedett karakter állhat. Ez az egész file-névre, illetve kiterjesztésre vonatkozik. Így a file-név kiterjesztése is - a pont helyét kivéve - tartalmazhat egy csillag karaktert. E két jelet globális név-jelnek hívjuk. Tartalomjegyzék-névben ezeket a karaktereket általában nem használhatjuk!

A globális név-karakterek szerepe egyes parancsok esetén más és más lehet. Van olyan eset, amikor az adott file-névvel azonosítható összes file-lal végrehajtódik az adott művelet, van, amikor csak az elsővel.

Példák:

A DIR parancs a lemez tartalomjegyzékét írja ki. Példánkban a lemez a következő file-okat tartalmazza:

```
COMMAND.COM
AUTOEXEC.BAT
ANSI.SYS
FORMAT.COM
CHKDSK.COM
SYS.COM
DISKCOPY.COM
DISKCOMP.COM
COMP.COM
EDLIN.COM
MODE.COM
FDISK.COM
BACKUP.COM
RESTORE.COM
PRINT.COM
RECOVER.COM
ASSIGN.COM
TREE.COM
GRAPHICS.COM
SORT.EXE
FIND.EXE
MORE.COM
BASIC.COM
BASICA.COM
KEYBUK.COM
KEYBFR.COM
KEYBSP.COM
KEYBIT.COM
KBDOS.BAS
KBPGM.BAS
WEDATIM.COM
GRAFTABL.COM
```

A DIR *.EXE és a DIR ????????.EXE parancsok hatása azonos. Ha a parancsok egyikét beadjuk, akkor a következő file-azonosítók jelennek meg a képernyőn:

```
SORT.EXE
FIND.EXE
```

A DIR C???????.* és a DIR C*.* parancsok hatása azonos. Ha a parancsok egyikét beadjuk, akkor a következő file-azonosítok jelennek meg a képernyőn:

```
COMMAND.COM
CHDSK.COM
COMP.COM
```

A DIR *.* parancs pedig valamennyi file-azonosítót kiírja.

2.5.3 Aktuális lemezegység

Az éppen aktuális lemezegység azonosítóját nem kell a file-névben szerepeltetni. Ez egy sor esetben kényelmessé teszi a DOS használatát, máskor (pl. FORMAT) használatának kellemetlen hatásai lehetnek. Indításkor az aktuális egység az, amelyikről a rendszert betöltöttük. A DOS lehetőséget biztosít az aktuális meghajtó megváltoztatására. Erre a d: parancs szolgál, ahol d valamelyik lemezegység betűjele.

Például:

C> (A kezdeti készenléti jel - a prompt. Tehát a C: jelű fix lemez az aktuális lemezegység.)

C>A: (Az új meghajtó kijelölése)

A> (Az A: lemezegység lett az aktuális meghajtó.)

Megjegyzés:

1. Hogy mi legyen a készenléti jel, a PROMPT paranccsal beállítható. (Lásd a 12. fejezetben a DOS speciális lehetőségeinek ismertetését.)
2. A perifériák hozzárendelésének megváltoztatására is van lehetőség a PC DOS rendszerben. Ez a SUBST (4.16), a JOIN (4.17) vagy az ASSIGN (6.7) paranccsal történhet.
3. Abban az esetben, ha megváltoztatjuk az aktuális lemezegység azonosítóját, akkor minden esetben az utoljára használt tartalomjegyzék lesz az aktuális.

Például ha a rendszert a C: egységről indítottuk el, akkor a következő parancsok hatására az aktuális tartalomjegyzék \TURBO lesz:

```
CD \TURBO
A:
CD \ADATOK
C:
```

Ha ezután újra kiadjuk az A: parancsot, akkor az aktuális tartalomjegyzék ismét \ADATOK lesz.

Egy lemezegység első használatakor a főtartalomjegyzék lesz az aktuális tartalomjegyzék.

2.5.4 Keresési sorrend

A DOS parancsok között két - alapvetően különböző - típus létezik. Az egyik típust belső, a másikat külső parancsnak hívjuk. A külső parancsok jellemzője, hogy a parancsnévvel azonos .EXE vagy .COM kiterjesztésű file-t a rendszer valamelyik háttértárolón keresi, onnan betölti, majd elindítja.

A rendszer több lemezegységet, azok több altartalomjegyzéket tartalmazhatnak. Alkalmanként pontosan meg kell adni, hogy a parancsot tartalmazó file melyikben is található. Erre több lehetőség is kínálkozik:

a) a parancsot megelőzően beírt elérési út megadásával:

[*d:*][*út*] *parancs paraméterek*

b) A PATH parancs (4.5) használatával.

Ha nem élünk az a), illetve b) lehetőségekkel, akkor a DOS a parancs nevével azonos (.EXE vagy .COM végződésű) file-t csak az aktuális lemezegység éppen használt (aktuális) tartalomjegyzékében keresi.

Az a) alkalmazása esetén a DOS a *d*: lemezegység "*út*" által kijelölt tartalomjegyzékében keresi a parancsnak megfelelő file-t. Ha a *d*: elmarad, akkor az aktuális lemezegység egyik tartalomjegyzékéről van szó. Ha az út megadása marad el, akkor a *d*: lemezegység utoljára használt tartalomjegyzékében fogja a DOS a file-t keresni. Ha a lemezegységet eddig még nem használtuk, ez a főtartalomjegyzék (root directory) lesz.

A PATH parancsban - b) eset - lehetőség van több elérési út megadására is. Ennek egyedül az szab határt, hogy egy DOS parancs hossza maximum 128 karakter lehet.

A B:\DOS\FORMAT parancs hatására a rendszer a B: lemezegység \DOS altartalomjegyzékében keresi először a FORMAT parancsnak megfelelő file-t. Ha a

PATH C:\WORK; D:\ADATOK

parancsot adtuk ki, akkor a rendszer a C: lemezegység \WORK, illetve a D: lemezegység \ADATOK altartalomjegyzékeiben keresi a kiadott parancsnak megfelelő file-t.

A fenti a)-b) sorrend egyben keresési sorrend is. A DOS először a parancs előtt álló [d:][út]-nak megfelelő, vagy ennek hiányában az aktuális lemezegység aktuális tartalomjegyzékében keresi a parancsnak megfelelő file-t, majd, ha itt nem találja, a PATH-ban megadott elérési utakat veszi sorba. Ha egyikben sincs, akkor hibajelzést kapunk.

A belső parancsok előtt nem használhatjuk a [d:][út] megadását. Ennek oka, hogy pl. DIR.EXE vagy DIR.COM nevű file nincs a lemezen.

2.6 A parancsok szintaxisa és a leírásuknál alkalmazott jelölésrendszer

A következő fejezettől kezdve konkrét DOS parancsokat ismertetünk. Elöljáróban megemlítünk néhány szabályt, amely a parancsok általános formájára vonatkozik, és bemutatjuk a parancsok leírásánál használt jelölésrendszert is.

A DOS parancsokat kis- és nagybetűkkel is írhatjuk. A könyv példáiban csak nagybetűket látunk, de ez az írásmód nem kötelező.

A billentyűzetről csak akkor adhatunk ki végrehajtandó DOS parancsokat, amikor a DOS prompt-ját (felszólító jelét) látjuk (pl. A> vagy C>). A parancs beírásának végén nyomjuk meg az <ENTER> billentyűt.

A DOS belső parancsait bármikor kiadhatjuk, de a külső parancsokat csak akkor, amikor az adott parancs programja a DOS számára elérhető (az aktuális, ill. a megnevezett meghajtó aktuális vagy pontosan megnevezett tartalomjegyzékében, vagy a PATH parancsban kijelölt úton található).

A DOS parancsok általános formája:

kulcsszó paraméterek

A *kulcsszó* határozza meg, milyen műveletet végezzen a DOS (például egy file törlését). A *paraméter*(ek) adja(ák) meg általában a művelet tárgyát (pl. a törlendő file nevét).

A *kulcsszót* a könyvben mindig vastag nagybetűvel írjuk, ezt betűről-betűre pontosan kell lemásolni. Nem minden kulcsszóhoz tartoznak paraméterek. A kulcsszó és az első paraméter között, valamint az egyes paraméterek között egy vagy több szóközt (space) kell hagynunk. (A szóköz helyett más elválasztó jelet: vesszőt, pontosvesszőt, egyenlőségjelet vagy a <TAB> billentyűt is használhatjuk. A könyvben - a jobb olvashatóság kedvéért - szóközt hagytunk az egyes elemek között.)

Előfordulhat, hogy egy parancs több paramétere közül nem mindegyik megadása kötelező. Ha pl. három paraméter közül csak az elsőt és a harmadikat akarjuk kiírni, akkor a szóköztől eltérő elválasztó jelet kell használnunk (pl. vesszőt). A hiányzó paramétert két egymás utáni vessző jelöli:

PARANCS PAR1,,PAR3

A kulcsszón kívül is mindent, amit itt vastag nagybetűvel adunk meg, változtatás nélkül kell beírni, ugyanez vonatkozik a speciális jelekre: pontokra, vesszőkre, csillagokra stb. is.

Amit a könyvben dőlt kisbetűvel jelölünk, azt nem szó szerint, hanem értelemszerűen, az aktuális értékkel helyettesítve kell beírni. Ha a parancsban a "*file-név*" szó szerepel, akkor ehelyett annak a file-nak a nevét kell beírni, amelyikre a parancsot alkalmazni akarjuk.

A szögletes zárójelek ([,]) között álló szövegeket nem kötelező beírni. A parancs leírásából kiderül, mi történik, ha az adott elemet megadjuk, ill. elhagyjuk. A szögletes zárójelek csak jelzések, a parancsok beírásakor ne használjuk azokat! (Vannak olyan esetek, amikor a szögletes zárójelek a parancs részei, s be kell azokat írni. Ezeket mindig külön megemlítjük.)

A függőleges vonallal elválasztott szövegrészek közül választanunk kell, és csak az egyiket kell beírnunk (mindkettőt nem szabad). A függőleges vonal sem része a parancsnak.

A három pont (...) azt jelenti, hogy az előtte álló elemet tetszés szerinti számban meg lehet ismételni.

A számítógép speciális billentyűinek nevét kisebb/nagyobb jelek közé zárva közöljük a szövegben (pl. <ENTER>, <F1> stb.). Ha tehát az <F1> jelöléssel találkozunk, ez azt jelenti, hogy nyomjuk le az F1 feliratú billentyűt, és nem azt, hogy az F és az 1 billentyűket használjuk.

3. DOS: file-kezelő parancsok

3.1 COPY - másolás

A parancs lehetővé teszi

- a file-ok másolását egy másik lemezre a file azonosítójának a megváltoztatása nélkül,
- a file-ok mozgatását ugyanazon a lemezen a file azonosítójának a megváltoztatásával (duplikálás),
- több file összemásolását egyetlenné
- a billentyűzetről, mint a rendszer input perifériájáról, adatok felvitelét a lemezre.

Verzió: 2.11-3.30
A parancs típusa: Belső
A parancs formája:

1. **COPY** [/A][/B][*d:*][*út*]*file-név*[*.kit*] [/A][/B][*d:*][*út*][*file-név*][*.kit*][/V]

azaz:

COPY [*paraméterek*] *forrás file-azonosító* [*paraméterek*] *cél file-azonosító*[/V]

2. **COPY** [/A][/B][*d:*][*út*] *forrás file-név 1* [*.kit*] +
[/A][/B][*d:*][*út*] *forrás file-név 2* [*.kit*] +
...
[/A][/B][*d:*][*út*] *cél file-név* [*.kit*][/V]

A paraméterek jelentése:

A /A és a /B paraméter a **másolás módját adja meg**. Minden paraméter, amit a parancssorba írunk, a következő /A vagy /B paraméterig valamennyi file-azonosítóra vonatkozik. (A /A és /B paraméterek csak a 3.x verziókban használhatók.) Ezeknek a paramétereknek jelentése a következő:

Jelentése a forrás file-ra:

/A	a program a file-t ASCII-kódú (szöveges) file-ként kezeli. A file-vége jelig (EOF) másolja a file-t, de azt már nem. (Az EOF <Ctrl-Z>, azaz a hexadecimális 1A.) Tehát több forrás file-t is össze lehet fűzni.
/B	a program az egész file-t másolja a tartalomjegyzékben lévő file-méretnek megfelelően.

Jelentése a cél file-okra:

/A	A file végére a rendszer kiteszi az EOF jelet.
/B	A cél file végére nem kerül EOF.
/V	A DOS ellenőrzi a másolat helyességét. Bár csak nagyon ritkán fordulnak elő lemezhibák a másolás során, mégis hasznos lehet ennek a paraméternek a megadása. Ekkor viszont a COPY parancs végrehajtási ideje megnő.

A /V paraméter hatása ugyanaz, mint a VERIFY ON parancsé. Ha a VERIFY ON-t beállítottuk, akkor felesleges a COPY parancsban a /V paraméter megadása. A különbség, hogy a /V csak a COPY végrehajtásának idején él, a VERIFY ON pedig a VERIFY OFF parancs kiadásáig.

Megjegyzés:

1. Ha sem az /A, sem a /B paramétert nem adtuk meg, akkora COPY maga választja meg,hogy melyiket használja. Ha a COPY első formáját használjuk, akkor ennek a paraméternek az alapértelmezése /B, ha file-ok összefűzését adtuk meg, akkor pedig /A.
2. Ha egy fenntartott egységnévről vagy egységnévre másolunk, akkor a /A ASCII-módban kerül végrehajtásra. Az első <Ctrl-Z> jel (EOF) leállítja a másolási folyamatot.
3. Ha egy olyan file-ról készítünk másolatot, amely csak olvasásra (READ ONLY) van kijelölve, akkor a másolat nem kapja meg ezt az attributumot.
4. A globális név-jelek használata (? és *) a paraméterekben megengedett. Ha ezeket használjuk a forrás file-oknál, akkor a másolás során az összes másolt file-nevet a rendszer kiírja a képernyőre.
   Ha a globális név-jeleket a cél file-nál adjuk meg, helyükre a megfelelő forrás file-név karakterei kerülnek. (pl.: COPY *.TXT *.OLD parancsnál az összes .TXT kiterjesztésű file-t átmásoljuk ugyanolyan névvel, de .OLD kiterjesztéssel.)
5. A COPY paraméterében megadott cél file-t, ha nincs olyan azonosítójú file (a megfelelő úton), a COPY hozza létre, ha van, akkor azt kérdés nélkül felülírja.
6. A PROPOS-16-ban a /M opció esetén az eredeti (forrás) file törlődik, /P paraméternél pedig egyenként engedélyezést kér az operációs rendszer, hogy az esetleg létező, azonos nevű file-t felülírja. A /V opció helyett PROPOS-16-ban a /C használatos.

A COPY parancsot három formában szokás használni:

I. Alkalmazás - Másolás ugyanazzal a névvel

Akkor alkalmazzuk, ha a forrás file-azonosítója megegyezik a cél file-éval. Ekkor a parancs formája:

COPY [*d*:][*út*]*file-név*[.*kit*]

vagy

COPY [*d*:][*út*]*file-név*[.*kit*] *d*:[*út*]

Az első formában az aktuális meghajtó aktuális tartalomjegyzékébe kell másolni. A második esetben cél-meghajtót és/vagy tartalomjegyzéket adunk meg. Mivel második file-azonosítót nem adtunk meg, így a cél file is ugyanazt a nevet kapja, ezért szükséges, hogy egy másik meghajtót is specifikáljunk, különben a másolás nem megengedett.

Példa:

1.1. Ha az A: az aktuális meghajtó, és a következő parancsot adjuk ki:

```
A>COPY C:SZOVEG.TXT
```

akkor a C: meghajtóról a SZOVEG.TXT azonosítójú file az A: meghajtóra kerül. A SZOVEG.TXT azonosító pedig bekerül az aktuális tartalomjegyzékbe.

1.2. Ha a következő parancsot adjuk ki:

```
A>COPY *.BAK B:
```

akkor az A: egységnek azok a file-jai, amelyek az aktuális tartalomjegyzékben vannak és .BAK a file-név kiterjesztésük, a B: meghajtó aktuális tartalomjegyzékébe másolódnak. Az átmásolt file-ok azonosítója megjelenik a képernyőn.

1.3. A következő parancsot a lemez sűrítésére, a file-ok átrendezésére is felhasználhatjuk. Az

```
A>COPY *.* B:
```

parancs hatására az A: meghajtó aktuális tartalomjegyzékéből valamennyi file a B: meghajtóra másolódik. Az átmásolt file fizikai blokkjai szekvenciálisan helyezkednek el a lemezen (feltételezve, hogy a B: meghajtóban üres lemez volt). Az átmásolt file-ok azonosítói megjelennek a képernyőn.

II. Alkalmazás - Másolás más névvel

Abban az esetben használjuk, ha a cél file neve más lesz, mint a forrás file-é. A parancs formája következő:

COPY [*út*]*file-név*[*.kit*] [*út*]*file-név*[*.kit*]
vagy
COPY [*út*]*file-név*[*.kit*] *d*:[*út*]*file-név*[*.kit*]

Az első formában egy file-t másoltunk és a másolat nevét megváltoztattuk. Meghajtót nem adtuk meg, ezért az aktuális meghajtót használtuk.

A második formában a file-t átmásoltuk és át is neveztük. Ebben a formában a célmeghajtót is megadtuk. Mivel a file nevét megváltoztattuk, a cél- és a forrásmeghajtónak nem kell különbözőnek lenni. Az aktuális tartalomjegyzék (az a tartalomjegyzék, amiben éppen vagyunk) is lehet ugyanaz.

Példa:

2.1. Ha a következő parancsot adjuk ki:

```
A>COPY SZOVEG.BAK B:*.TXT
```

akkor az A: meghajtóról a SZOVEG.BAK azonosítójú file a B: meghajtóra másolódik. A másolat azonosítója a SZOVEG.TXT lesz.

2.2. A másolás során a fenntartott egységnevek is használhatók. Például:

```
COPY CON: ADATOK.TXT
COPY CON: AUX
COPY CON LPT1
COPY ADATA CON:
COPY ADATB AUX:
COPY ADATC LPT2:
COPY AUX: LPT1:
COPY AUX: CON
```

A NUL egységnév is használható. Ha fenntartott egységnevet használunk, akkor nem szükséges a kettőspont (:) megadása. Ennek ellenére - a jobb olvashatóság miatt - célszerű kitenni. Így nem keverhetjük össze file névvel.

2.3. Ebben a példában a COPY parancs segítségével a billentyűzetről viszünk be adatokat és lemezen létrehozunk egy ADATOK.TXT azonosítójú file-t. A parancs a következő:

```
A>COPY CON: ADATOK.TXT
```

Ezután a szöveg beírása következik soronként. A sorok végét az üzenetvégét jelző billentyű, az <ENTER> jelzi.

Most új sor következik.
.
.
.

Ha befejeztük a szöveg beírását, akkor az <ENTER> után a <Ctrl-Z>-t (és utána <ENTER>-t) kell beírni. Ez írja fel a file-ra a file-vége jelet (EOF).

III. Alkalmazás - File-ok másolása és összefűzése

Akkor használjuk, ha a másolás során a file-okat össze akarjuk fűzni. Ez azt jelenti, hogy egy file-hoz egymás után több file is hozzácsatolható úgy, hogy mindegyik a megelőző file utolsó rekordja után kerül. Az új file file-leíró blokkjában (a file control block-ban) a dátum és az idő a napi dátum és idő, az a mező pedig, amely a file hosszát adja meg, az eredmény file-ra vonatkozik.

COPY [/A][/B][*d*:][*út*]*forrás file-név 1* [*kit*] +
[/A][/B][*d*:][*út*]*forrás file-név 2* [*kit*] +
...
[/A][/B][*d*:][*út*] *cél file-név* [*kit*][/V]

Az összefűzésre kerülő file azonosítókat a COPY parancs paraméterei között fel kell sorolni. Az egyes input file-okat plusz jel (+) választja el egymástól.

Itt a parancsnak a szakasz elején ismertetett második formáját kell használni.

Példa:

3.1. Ha a következő parancsot adjuk ki:

```
A >COPY SZOVEG1.TXT + SZOVEG2.TXT SZOVEG.TXT
```

akkor az aktuális meghajtón a SZOVEG.TXT file-t a SZOVEG1.TXT és a SZOVEG2.TXT file-ok összekapcsolásával állítjuk elő.

Megjegyzés:

1. Ha a cél-file azonosítóját nem adjuk meg, akkor a forrás file-ok közül az első azonosítóját tartja meg.
2. A file-ok összefűzésének alapértelmezése a szöveg mód (ASCII kód). Ez azt jelenti, hogy az első EOF jelet (<Ctrl>-Z, azaz hexadecimális 1A) a file végének tekinti.
   A bináris formájú file-ok összefűzésére a /B paramétert kell használnunk. Ez lehetővé teszi, hogy a COPY a teljes file-t átmásolja.
   Az ASCII- és a bináris formájú file-ok összefűzhetők, de akkor a /A és a /B paramétereket értelemszerűen kell használni.
   A /A vagy a /B arra a file-ra vonatkozik, amely előtt áll, és mindaddig vonatkozik az azt követő file-okra, amíg másik /B vagy /A paraméter nem következik.
3. A globális file-név jelek (a * és a ?) a file-azonosítókban megadhatók. A COPY parancs akkor használható a legegyszerűbben, ha a file-ok azonosítóit jól választottuk meg és a globális file-név jelek segítségével tudjuk kijelölni a másolandó file-okat.
4. A file összefűzési eljárás akkor eredményes, ha a COPY legalább egy forrás file-t megtalált. Ha egyetlen file-t sem talált meg, akkor az üzenet a következő:

   ```
   0 DATEI(EN) KOPIERT
   ```

   illetve

   ```
   0 FILE(S) COPIED
   ```

3.2 XCOPY - file-ok csoportos másolása

A parancs lehetővé teszi file-ok és altartalomjegyzékben szereplő file-ok különböző szempontok szerinti kiválogatását, majd másolását.

Verzió: 3.20-3.30
A parancs típusa: Külső
A parancs formája:

XCOPY [*d*:][*út*][*file-név*[*.kit*]][*d*:][*út*][*file-név*[*.kit*]][/A][/D][/E][/M][/P][/S][/V][/W]

A paraméterek jelentése:

Az első [*d*:][*út*][*file-név*[*.kit*]] a forrás helyét jelöli ki. Vagy a [*d*:], vagy az [*út*], vagy a [*file-név*[*.kit*]] résznek szerepelnie kell. A *file-név*[*.kit*] globális név-karaktereket is tartalmazhat. Ha a *file-név*[*.kit*] elmarad, a teljes tartalomjegyzék tekintendő forrásnak.

A második [*d*:][*út*][*file-név*[*.kit*]] a cél helyét jelöli ki. Ez az a tartalomjegyzék (vagy annak a *file-név* [*.kit*]-tel megadott része), ahová a másolás történik. A *file-név*[*.kit*] tartalmazhat globális név-karaktereket. A file-név megadása csak az átmásolt file-ok átnevezésére szolgál.

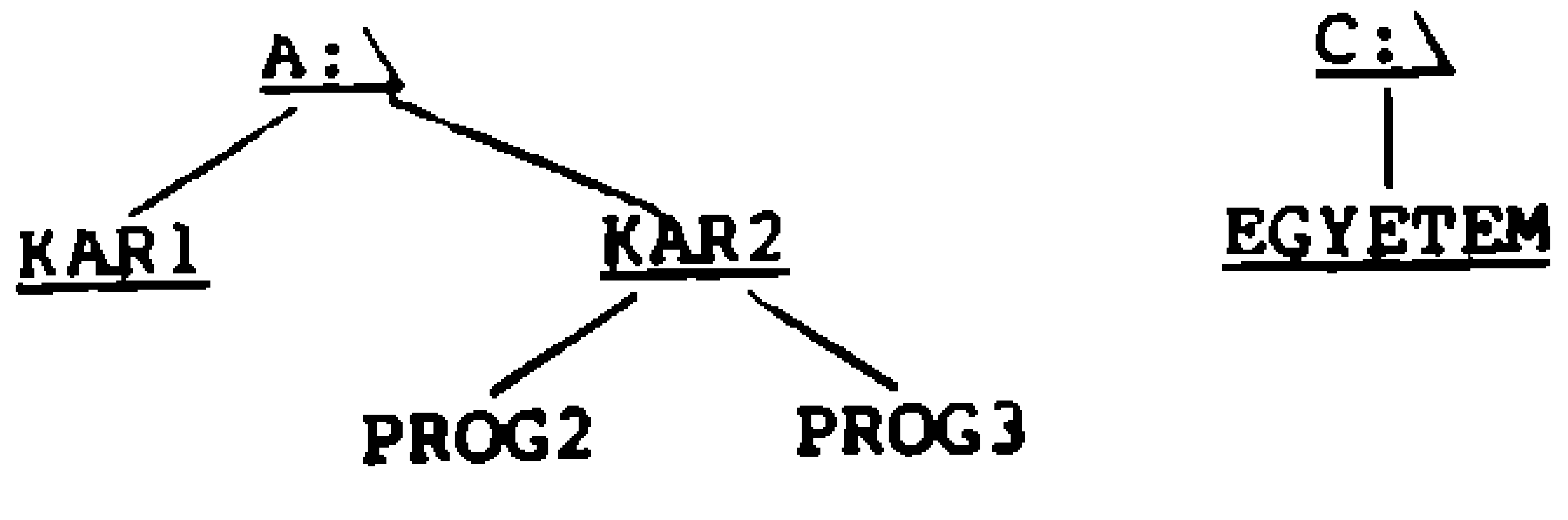
/A	Csak azok a file-ok kerülnek átmásolásra, amelyek archív attributum bitje beállított. A forrás file-ok archív attributum bitje nem módosul.
/D	A dátum megadja, hogy csak az ekkor vagy ezután módosított file-okat kell a parancsnak átmásolnia. A dátumot a SELECT vagy a COUNTRY parancsban megadott formátumnak megfelelően kell megadni. Például így: /D:88-01-23.
/E	A parancs azokat az (al)tartalomjegyzékeket is létrehozza a cél tartalomjegyzékben, amelyek (a kiválasztás után) üresek maradnak.
/M	Csak azok a file-ok kerülnek átmásolásra, amelyek archív attributum bitje beállított. Az XCOPY a másolás után ezen file-ok archív bitjét törli.
/P	Minden egyes kiválasztott file-ról külön is megkérdezi, hogy átmásolja-e. Ha Y-nal válaszolunk, a másolás megtörténik, ha N-nel, akkor nem.
/S	A paraméter hatására a forrás tartalomjegyzék altartalomjegyzékét is (teljes mélységben) átmásolja. Az üres, de létező tartalomjegyzékek létrehozására csak a /E paraméter használatával együtt kerül sor.
/V	A parancs ellenőrzi a másolás helyességét.
/W	A parancs megkezdése előtt az XCOPY a következő üzenetet küldi: Press any key to begin copying file(s) Lehetőség van másik lemez behelyezésére, majd egy tetszőleges billentyű megnyomására megkezdődik a másolás.

Megjegyzés:

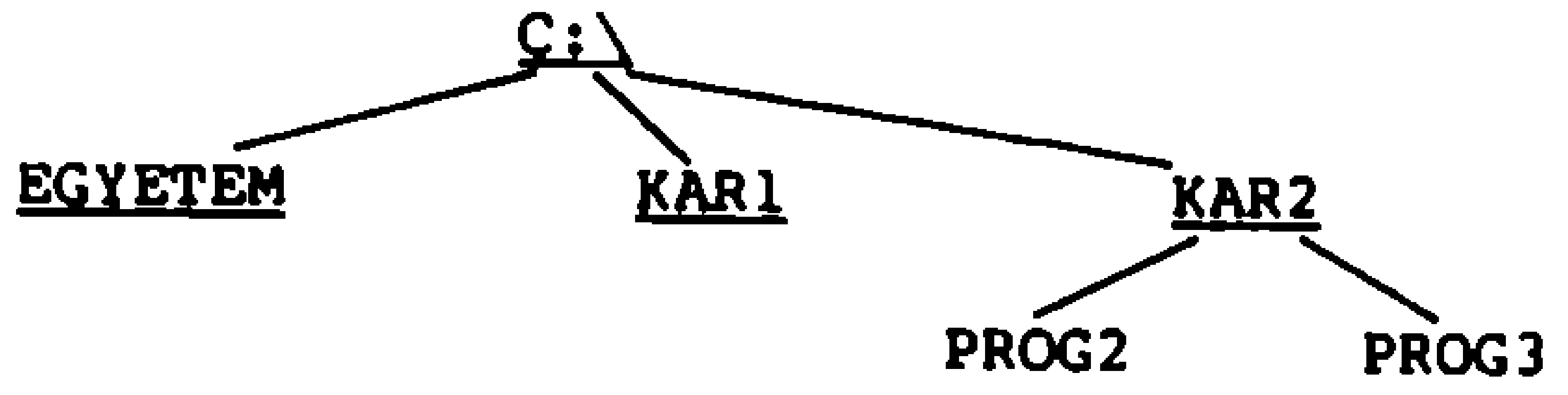
1. A kezdeti tartalomjegyzék és file-név a fő-tartalomjegyzék, illetve a *.* file-név.
2. A másolat file-ok neve is megadható, s ezzel a másolat átnevezhető.
3. Logikai egységekről vagy egységekre (CON, LPT1 stb) nem másolhatunk az XCOPY-val.
4. Az XCOPY a DOS 3.20-nál korábbi verzióiban nem használható.

Példa:

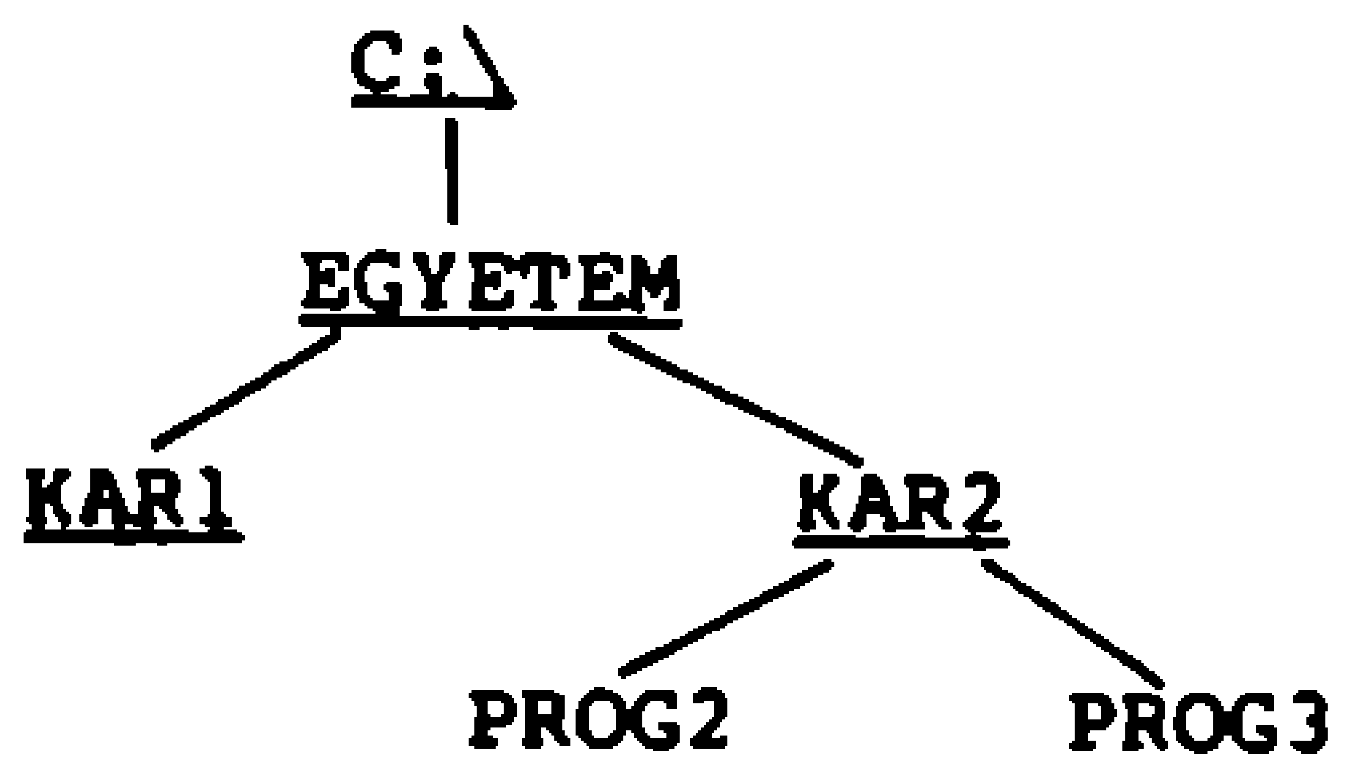
Tegyük fel, hogy az A: és C: lemezek tartalomjegyzéke az alábbi:



Az XCOPY A:\ C:\ /S parancs kiadása után a C: lemez tartalomjegyzéke így alakul:



Ha viszont az XCOPY A:\ C:\EGYETEM /S parancsot adjuk ki a C: egység tartalomjegyzéke így alakul:



3.3 REPLACE - file-ok cseréje

A parancs lehetővé teszi, hogy a célon lévő file-okat a forráson lévő kijelölt file-okkal helyettesítsük.

Verzió: 3.20-3.30
A parancs típusa: Külső
A parancs formája:

REPLACE [*d*:][*út*]*file-név*[*.kit*] [*d*:][*út*][**/A**][**/P**][**/R**][**/S**][**/W**]

A [*d*:][*út*]*file-név*[*.kit*] adja meg a forrás file-t, amit a célon helyettesíteni kell /vagy hozzáadni/. A *file-név*[*.kit*] tartalmazhat globális név-karaktereket is. A második [*d*:][*út*] egy cél lemezegységet és annak egy tartalomjegyzékét jelöli ki. Az ebben lévő file-okat kell helyettesíteni, vagy a /A paraméter használata esetén ide kell másolni a forrás file-okat.

A további paraméterek jelentése a következő:

/A A REPLACE a forrásból átmásolja a célra az összes olyan file-t, ami nem létezik a cél tartalomjegyzékben.

/P A parancs minden egyes kiválasztott file után megkérdezi, hogy át akarjuk-e másolni. Y válasz esetén a másolás megtörténik, N válaszra nem.

/R A parancs a cél tartalomjegyzékben lévő, csak-olvasható attributumú file-okat is felülírja.

/S A REPLACE a cél tartalomjegyzék összes altartalomjegyzékét végigkeresi a helyettesítendő file-ok után kutatva.

/W A parancs végrehajtása előtt a meghajtóba kéri a forrás lemezt, pontosabban a következő üzenet jelenik meg:

```
Press any key to begin replacing file/s/ vagy
Press any key to begin adding file/s/ (/A használatakor)
```

Példa:

```
REPLACE C:\MUNKA\PELDA.TXT A:\ /S
```

A parancs a \MUNKA altartalomjegyzékben lévő PELDA.TXT nevű file-t megkeresi az A: egységen akármelyik altartalomjegyzékben (/S), majd helyettesíti.

Megjegyzés:

1. A /A és /S paraméter nem használható együtt.

2. A parancs az alábbi hibakódokkal tér vissza:

Kód	Hibaüzenet
2	File not found (a file nem létezik)
3	Path not found (az út nem létezik)
5	Access denied (az elérés tilos)
8	Insufficient memory (nincs elegendő memória)
11	Invalid format (érvénytelen formátum)
15	Invalid drive was specified (nem létező lemezegységet adtunk meg)
22	Unknown command (ismeretlen parancs; csak 3.2-nél korábbi verzióban!)

3. A DOS 3.2-nél korábbi verziói nem tartalmazzák ezt a parancsot.

3.4 COMP - összehasonlítás

A COMP (COMPARE, összehasonlítás) parancs lehetővé teszi, hogy file-okat byte-onként összehasonlítsunk. Általában a COMP parancsot a COPY /V parancs helyett használjuk. Meggyőződhetünk vele arról, hogy a másolás művelete sikeres volt-e.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

COMP [*d*:][*út*][*file-név*[.*kit*]] [[*d*:][*út*][*file-név*[.*kit*]]

A paraméterek jelentése:

A COMP paranccsal két file hasonlítható össze. Az első parancs paraméter az elsődleges file-t, a második pedig a másodlagost határozza meg. Az összehasonlításra kerülő file-ok azonos lemezen is lehetnek.

A globális név-jelek használata mindkét file-névben megengedett.

Ha a parancsnál nem, vagy hiányosan adunk meg paramétert, akkor a COMP üzenetet küld, és kéri a hiányzó paraméter megadását.

A COMP működése

A parancs először a megadott két file méretét hasonlítja össze. A file-ok méretét a tartalomjegyzékből olvassa ki.

Ha a két méret különbözik, akkor üzenetet küld, és befejezi működését.

Abban az esetben, ha a két méret megegyezik, elkezdi a file-ok elejétől byte-onként az összehasonlítást. Ha a két byte különbözik, akkor hibaüzenetet kapunk. A hibaüzenet tartalmazza a byte-ok címét a file elejéhez viszonyítva és az eltérő két információt hexadecimális formában.

Például:

```
Compare error at offset 17A
File 1 = 6E
File 2 = 6D
```

Tíz hibás összehasonlítás után a COMP nem folytatja az összehasonlítást. Befejezi a tevékenységét, és erről a következő üzenetet küldi:

```
10 Mismatches - ending compare
Compare more files (Y/N)?
```

Ha a válasz N, a COMP befejezi működését.

A COMP a file-ok összehasonlítását úgy fejezi be, hogy a file-vég-jel meglétét is ellenőrzi (EOF, hexadecimális 1A).

Ha ezt nem találta meg, akkor erről is üzenetet küld. Ez általában bináris file-oknál fordul elő, melyek végét - a szöveg-file-okkal ellentétben - nem jelzi külön EOF byte. Az üzenet bináris file-ok összehasonlítása esetén figyelmen kívül hagyható. Ha az összehasonlításnál nem talált hibát, akkor a következő üzenetet kapjuk:

```
Files compare ok
```

Ha a COMP a file-ok összehasonlítását befejezte, üzenetet küld, hogy további file-ok összehasonlítását el kell-e végeznie. Az üzenet a következő:

```
Compare more files (Y/N)?
```

Ha a válaszunk N, akkor a COMP befejezi működését. Ha viszont Y a válasz, akkor a COMP megkérdezi az elsődleges és a másodlagos file azonosítóit.

Például:

```
Compare more files (Y/N)?y
Enter primary file name
autoexec.bat

Enter 2nd file name or drive id
autoexec.1
```

Példa:

1. Ha például a következő parancsot adjuk meg:

```
C>COMP A:*.TXT B:*.BAK
```

ez azt eredményezi, hogy az A: meghajtón lévő valamennyi .TXT kiterjesztésű file összehasonlításra kerül a B: meghajtón lévő azonos nevű, de .BAK kiterjesztésű file-lal.

Az összehasonlításra kerülő file-ok azonosítóit kijelzi.

2. Ha két tartalomjegyzék file-jait szeretnénk összehasonlítani, akkor a következőket kell megadni:

```
A>COMP C:\BEJEGYZ1 C:\BEJEGYZ2
```

Ez azt eredményezi, hogy a C:meghajtó BEJEGYZ1 tartalomjegyzékének valamennyi file-ját összehasonlítja a BEJEGYZ2 tartalomjegyzék megfelelő file-jával.

Ha a COMP nem talál egyetlen megegyező azonosítójú file-t sem, akkor ismételten bekéri az elsődleges és másodlagos paramétereket.

Megjegyzés:

1. Mindkét file-nak lehet ugyanaz az azonosítója és még az elérési útja is, feltéve, hogy különböző lemezen találhatók.
2. A COMP parancs nem vár arra, hogy az összehasonlításra kerülő file-okat tartalmazó lemezeket elhelyezzük a meghajtókon. Ezért a COMP kulcsszó megadása után várjuk meg, míg a program a file nevére rákérdez, és csak ezután cseréljük le a lemezt.
3. A PROPOS-16 operációs rendszerben a COMP parancson kívül még a következő parancsok végzik a file-ok összehasonlítását:

   ```
   CMP [d:][út]file-név[.kit] [d:][út]file-név[.kit][/L][/S]
   ```

   Ha a /L opciót adjuk meg, akkor a két file valamennyi eltérő byte-ját kilistázza hexadecimális alakban.

   Ha a /S opciót írjuk be, akkor csak a visszatérési kódot állítja be a következőképpen:
   0, ha a két file megegyezik;
   1, ha a két file különböző;
   2, ha a file-ok egyikét nem lehet megnyitni.

   Ha nem adunk meg opciót, akkor csak az első nem egyező byte-okat írja ki.

   ```
   COMP [d:][út]file-név[.kit] [d:][út]file-név[.kit][/123]
   ```

   Két karakteres file-t hasonlít össze soronként. A következő formában küldi az üzenetet:

   1. oszlop: csak az elsőként megadott file-ban szereplő sorok;
   2. oszlop: csak a második file-ban szereplő sorok;

3. oszlop: azok a sorok, amelyek mindkét file-ban megtalálhatók.

Az opciók megadásával lehet kiválasztani, hogy mely oszlopokat írja ki az outputra. Az opciók tetszés szerinti kombinációban használhatók. Például érvényesek a következő opciók: /12 és /13 vagy /1 /2 /3.

Ha nem adunk meg opciót, akkor nem jelenik meg üzenet. A file-oknak nem kell rendezettnek lenni, és méretük is különbözhet.

4. PROPOS-16 operációs rendszerben további parancsok segítik a karakteres file-ok kezelését.

A DIFF parancs közli, hogy melyek azok a változtatások, amelyek ahhoz szükségesek, hogy az elsőként megadott file megegyezzen a másodikként megadottal. A szükséges változtatásokat soronként közli.

A formája:

```
DIFF [d:][út]file-név[.kit] [d:][út]file-név[.kit][/E][/B]
```

Az opciók jelentése:

Alapértelmezés, hogy az első file-ra vonatkozó sorszámmal közli a szükséges változtatást:

I - INSERT (Beszúrás a megadott sorszám elé.)
D - DELETE (Törlés).

A /E hatására a változtatások listája a sorszerkesztő (EDIT) szintaktikájában készül. Ez közvetlenül a sorszerkesztő inputjára is adható.

```
DIFF ADAT1 ADAT2 /E: EDIT JAVÍTOTT.TXT
```

A /B opció törli a sorok kezdetén lévő üres (blank) karaktereket és a tabulátor karaktert is.

Gazdaságosan használható programfejlesztésnél, mivel a forrás file-nál jóval rövidebb lesz a DIFF file, az üres karaktereket töröljük.

Ha a DIFF parancs nem tudja megnyitni valamelyik file-t, akkor hibaüzenetet küld. File-csoportok megadása nem lehetséges.

3.5 ERASE (DEL) - törlés

A parancs lehetővé teszi megadott file-ok törlését a tartalomjegyzékből.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

ERASE [*d*:][*út*][*file-név*[.*kit*]]
DEL [*d*:][*út*][*file-név*[.*kit*]]

Megjegyzés:

1. Mindkét parancsnál vagy a file-névnek, vagy az útmegadásnak szerepelnie kell.
2. Az ERASE szinonímája a DEL kulcsszó.
3. Globális név-jelek is használhatók. Ha az aktuális tartalomjegyzékből összes file-t törölni akarjuk, akkor a

   ```
   DEL [d:]*.*
   ```

   parancsot kell beadni.

   Ha ezt a kijelölést használjuk, akkor a végrehajtás előtt a rendszer megkérdezi, hogy valóban minden file-t törölni szeretnénk-e:

   ```
   A >del *.*
   Are you sure (Y/N)?
   ```

   Biztos benne? (I/N).

   Csak akkor történik meg a törlés, ha Y-t válaszolunk.

   Az IBMBIO.COM és az IBMDOS.COM file-okat és általában a rejtett (hidden) vagy rendszer (SYS) attribútum file-okat, valamint az altartalomjegyzékeket nem lehet ezzel a paranccsal törölni.
4. A szerkesztő-programok egy része a szöveg-file új változatának lemezre írásával egyidőben a régi változatot .BAK kiterjesztésűre változtatja. Ezek egy idő után feleslegessé válnak. Ezeket törölhetjük a C > DEL *.BAK parancs segítségével.

Példa:

1. Ha a következőt írjuk be:

```
C > DEL A:PROGRAM.BAS
```

a DOS törli az A: egységen levő PROGRAM.BAS programot.

2. Ha a következőt írjuk be:

```
C > DEL \MYDIR
```

és \MYDIR egy altartalomjegyzék, akkor a DOS visszakérdezés után törli az összes, a \MYDIR-ben levő file-t.

3.6 RENAME - átnevezés

A parancs file-ok átnevezésére szolgál.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

REN[AME] [*d*:][*út*]*file-név*[*.kit*] *file-név*[*.kit*]

A parancs működése:

Az első paraméterben megadott file azonosítóját a második paraméterben megadottra változtatja. A file-név továbbra is ugyanabban a tartalomjegyzékben marad.

Megjegyzés:
1. Az útmegadás csak az első paraméternél használható.
2. A paraméterekben a globális név-jelek is használhatók.

Példa:

Ha a következőket adjuk meg:

```
REN B:\ALTART1\PROG1.TXT PROG2.TXT
```

akkor az ALTART1 azonosítójú (al)tartalomjegyzékben lévő PROG1.TXT file-t átnevezi, és a PROG2.TXT az új file-név, ami szintén az ALTART1 altartalomjegyzékben van. Ha a

```
REN A:*.TXT *.LEV
```

parancsot adjuk ki, az A: lemez aktuális tartalomjegyzékében lévő összes .TXT kiterjesztésű file kiterjesztését .LEV-re változtatjuk.

3.7 RECOVER - helyreállítás

A parancs lehetővé teszi, hogy a meghibásodott szektorokban található file-okat a hibás területen kívül ismételten előállítsuk.

Verzió: 2.11-3.30
A parancs típusa: Külső
A parancs formája:

RECOVER [*d*:][*út*]*file-név*[*.kit*]

vagy

RECOVER *d*:

A paraméterek jelentése:

Az első formában csak egy file-t fog a rendszer ismételten előállítani. A második formáját akkor használjuk, ha a tartalomjegyzék sérült meg, ekkor a rendszer a lemez valamennyi file-ját ismételten előállítja.

A RECOVER abban az esetben, ha egy file ismételt előállítására használjuk, akkor a file-t szektorról szektorra átmásolja, a hibás részeket átlépi. A file-név nem változik.

Ha a tartalomjegyzék hibás, akkor valamennyi file-t ismételten előállítja, mivel nem tudja megállapítani, hogy melyik bejegyzés a hibás. Ez a file-leíró táblázat (FAT) alapján történik. A file-okat a következő névvel látja el:

```
FILEnnnn.REC
```

ahol nnnn a sorszámot jelöli, kezdőértéke: 0001.

Példa:

Ha egy lemez tartalomjegyzéke megsérült, és szeretnénk a tartalmát ismételten előállítani a B: meghajtón, akkor ezt a következő formában tehetjük:

```
RECOVER B:
```

A parancs beadása után a rendszer a lemez valamennyi file-ját ismételten előállítja. Az új file-nevek a következők:

```
FILE0001.REC
FILE0002.REC
.
.
.
```

Megjegyzés:

1. Ha a globális név-jeleket használjuk, a RECOVER akkor is csak egy file helyreállítását végzi el.
2. Szöveges file-ok ismételt előállítása után rendszerint még egy szövegszerkesztésre is szükség van. A file végéről a felesleges adatokat célszerű törölni.
3. Az első formánál, ha a file futtatható program (.COM illetve .EXE) volt, a hibás szektorok kihagyása miatt a visszanyert alak általában már használhatatlan programot eredményez!
4. Vigyázzunk! Ha a második formát használjuk, a parancs a lemezen levő - jó - file-okat is átnevezi FILExxxx.REC-re. Emiatt a parancs kiadása előtt mindenképpen mentsük el a még használható file-okat!
5. Vigyázat! A parancs a lemezen levő altartalomjegyzékeket egy-egy file-ba sűríti (a FAT alapján), így a bennük levő file-ok szétválogatása eléggé kilátástalan munka.

3.8 ATTRIB - File attributumainak beállítása

A parancs lehetővé teszi, hogy beállítsuk vagy lekérdezzük egy file archív vagy csak-olvasható attributum bitjét.

Verzió: 3.0 - 3.30
A parancs típusa: Külső
A parancs formája:

ATTRIB [+R][-R][+A][-A][*d*:][*út*]*file-név*[.*kit*]

A parancs a [d:][út]file-név[.kit] file archív vagy csak-olvasható attributum bitjét állítja be vagy kérdezi le. Ha a +R/-R, illetve +A/-A paraméterek egyikét sem adjuk meg, akkor a parancs kiírja a file archív, illetve csak-olvasható attributum bitjét.

Az egyes paraméterek hatása a következő:

+ R	a file csak-olvasható lesz;
– R	a file csak-olvasható attributum bitje törlődik;
+ A	a file archív file lesz;
– A	a file archív attributum bitje törlődik.

Megjegyzés:

1. Az archív attributum bitnek a BACKUP /M, illetve XCOPY /M parancsok használatakor van jelentősége.
2. A file-név[.kit] megadásakor használhatunk globális név-karaktereket (*, ?).

Példa:

```
ATTRIB +R -A KON4.TXT
```

A parancs hatására a KON4.TXT file csak-olvasható lesz, s az archív attributum bit törlődik.

3.9 SHARE - file megosztási környezet kezelése

A parancs betölti a file megosztási környezetet, ami lehetővé teszi a file-ok írásának és olvasásának ellenőrzését. Részletesen erről a 3DH és 3CH sorszámú DOS funkciók tárgyalásakor szólunk a 17. fejezetben.

Verzió: 3.0 - 3.30
Parancs típusa: Külső
Parancs formája:

SHARE [/F:*file méret*][/L:*zárolások*]

A paraméterek jelentése:

/F	file méret megadja az írási/olvasási műveletek ellenőrzéséhez szükséges memória terület nagyságát. Minden megnyitott file 22 byte-ot foglal el. A kezdeti érték 2048.
/L	zárolások megadja a használni kívánt zárolások számát. Kezdeti értéke 20.

Megjegyzés:

1. Ha a SHARE parancsot másodszor kíséreljük meg kiadni, akkor "Share Already Installed" üzenetet kapunk.
2. Egyes lemezegységek képesek annak ellenőrzésére, hogy kicseréltük-e benne a lemezt vagy sem. Ha ilyen történik egy írási/olvasási művelet alatt, a SHARE ellenőrzi, hogy jó lemez van-e bent, majd megismétli a műveletet.
3. A SHARE parancs a DOS 3.0-ásnál korábbi verzióiban nem használható.

3.10 TYPE - kiíratás

A parancs lehetővé teszi, hogy a file tartalmát a képernyőre kiírjuk.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

TYPE [*d*:][*út*\]*file-név*[*.kit*]

A parancs működése:

A megadott file tartalma megjelenik a képernyőn. A karakteres file-ok olvasható formában kerülnek kijelzésre. A bináris file-ok esetén ez a megjelenítés általában nem sok információt ad a file-ról.

A képernyő tartalmát nyomtatóra is kiírathatjuk a <Ctrl-PrtSc> billentyűk lenyomásával. Az outputot a nyomtatóra vagy egy file-ba is átirányíthatjuk. (Ezt a 12. fejezetben ismertetjük "A standard I/O átirányítása" című szakaszban.)

Megjegyzés:

1. A globális név-karakterek nem használhatók. Ha mégis megpróbálkozunk velük, akkor a következő üzenetet kaphatjuk:

   FILE NOT FOUND
   (A file nem található.)

2. A szöveges file-ban lévő tabulátor-karaktereket a TYPE felismeri. A tabulátor után következő karaktert a megfelelő tabulátor-pozícióba helyezi. (A tabulátor pozíciók: 8., 16., 24., 32. oszlop és így tovább.)
3. A PROPOS-16 operációs rendszerben a TYPE parancs paraméterek nélkül is megadható. Ekkor a rendszer a további parancs-paramétereket a standard input perifériáról várja.
4. A file képernyőre listázása megállítható a <Ctrl-S> vagy a <Ctrl-NumLock> billentyű megnyomásával. Ilyen esetben a képernyőről a <Ctrl-PrtSc> használatával másolat készíthető a nyomtatóra. A listázás tetszőleges billentyű megnyomására folytatható. A <Ctrl-C> megnyomása megszakítja a parancs végrehajtását.

Példa:

1. Az AUTOEXEC.BAT file tartalmát például a következő parancs kiadásával olvashatjuk el:

```
C>TYPE AUTOEXEC.BAT
```

2. Ha a file tartalmát ki akarjuk a nyomtatóra íratni, ezt a standard output átirányításával érhetjük el: C>TYPE PROGRAM.PAS >PRN:

A fenti parancs hatására a PROGRAM.BAS file tartalma a nyomtatóra kerül.

Hosszú szövegeket kényelmesebben olvashatunk, ha képernyőnként íratjuk ki a szövegeket. Ezt például a fenti 'PROGRAM.BAS' file-lal a következőképpen tehetjük meg:

```
C>TYPE PROGRAM.BAS‡ MORE
```

3.11 PRINT - nyomtatás

A parancs lehetővé teszi, hogy a nyomtatón listákat készítsünk, miközben a számítógép más feladatokat is ellát.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

PRINT [/D:*egységnév*][/B:*pufferméret*][/U:*foglalt*][/M:*nyomtat*][/S:*várakozik*]
[/Q:*sorméret*][/C][/T][/P][*d*:][*út*][*file-név*.[*kit*]...]

A PRINT működése

A parancs **spooling** funkciót valósít meg. A parancs első kiadása után a rendszer mérete kb. 3200 byte-tal megnövekszik. Ezen a memóriaterületen, mintegy "BACKGROUND" partícióban működik a PRINT, miközben a számítógépen tovább dolgozhatunk.

A program rendelkezik egy várakozási sorral, amelyben legfeljebb 32 file-nevet tud tárolni. A PRINT paranccsal lehet a várakozási sort kezelni. A sorhoz hozzáadhatók újabb file-nevek, törölhetők belőle, de a teljes várakozási sort is lehet törölni.

A paraméterek jelentése:

/T	hatására a várakozási sor valamennyi eleme törlődik,
/C	segítségével a megadott file-okat töröljük a várakozási sorból,
/P	segítségével új file-neveket fűzhetünk fel a várakozási sorba. (Ez az alapértelmezés.)
/B:*pufferméret*	segítségével a PRINT belső pufferének hosszát adhatjuk meg. Ezt a PRINT 512-nek választja meg. A nagyobb pufferméret hatékonyabbá teszi a parancs használatát.
/U:*foglalt*	azoknak az óraciklusoknak a száma, ameddig a PRINT-nek várakoznia kell, míg a nyomtató elérhető lesz ismét. A PRINT ezt az értéket 1-re állítja be. Az U: foglalt értéknek az 1-255 intervallumba kell esnie.
/M:*nyomtat*	azoknak az óraciklusoknak a száma, amennyi idő alatt a nyomtató ki tud nyomtatni egy karaktert. A PRINT ezt 2-re állítja be; lehetséges értéke az 1-255 intervallumba kell, hogy essen.
/S:*várakozik*	a várakozási időszak számát adja meg. Ha a rendszerben nem állítottuk be a végtelen sokszori kísérletezést, akkor ennyiszer próbálkozik a PRINT, mielőtt úgy dönt, hogy a nyomtató egység használhatatlan. A PRINT ezt az értéket kezdetben 8-ra állítja be. A lehetséges érté-

	kek: 1-255.
/Q:*sorméret*	a várakozási sor nagyságát adja meg. Alapértelmezése 10, értéke 1 és 32 közötti egész szám lehet.
/D:*egységnév*	a PRINT által használandó eszköz nevét adja meg. Ha a parancs első alkalmazásakor nem adjuk meg ezt a paramétert, a PRINT megkérdezi, hogy melyik egységet akarjuk használni. A lehetséges válaszok: LPT1, LPT2, LPT3, PRN, COM1, COM2, AUX.

Megjegyzés:

1. A /D, /B, /Q, /S, /U és /M paramétereket csak a parancs első hívásakor használhatjuk.
2. A DOS 2.10 és 2.11-es verziójában csak a /C, /T és /P paraméterek használhatók.
3. A PRINT használata mellett más módon (pl. <PrtSc> vagy <Ctrl-P>) nem használható a kijelölt periféria. Ha ezzel kísérletezünk, "out-of-paper" hibaüzenetet kapunk.
4. A parancssorban több file-név is megadható. Az azonosítókban a globális név-jelek is használhatók.
5. Csak az aktuális tartalomjegyzékben szereplő file-neveket lehet egy várakozási sorban megadni.
   A várakozási sor megadása után az aktuális tartalomjegyzék változtatható anélkül, hogy a várakozási sort ez érintené.
6. A kinyomtatandó file-okat tartalmazó lemezt mindaddig a megadott meghajtóban kell hagyni, amíg a teljes nyomtatási folyamat be nem fejeződik.
7. Ha a DOS a nyomtatás folyamán a lemez olvasása során hibát észlel, akkor a következő történik:
   - A nyomtatás alatt álló file-t törli a várakozási sorból.
   - Hibaüzenetet küld a nyomtatóra.
   - A nyomtató új lapot kezd és hangjelzést is ad.
   - A várakozási sorban következő file kinyomtatását elkezdi.
8. Ha a /T vagy a /C paramétert nyomtatás alatt álló file esetében adjuk meg, akkor a következő történik:
   - A nyomtató hangjelzést ad.
   - A nyomtatón üzenet jelenik meg a törölt file-(ok)ról.
   - A nyomtató új lapot kezd.
   - Ha a várakozási sorban még van file, akkor annak a nyomtatásával folytatódik a kiírás.

Példa:

1. Tételezzük fel, hogy a PRINT első használatakor a következő parancsot adtuk ki.

```
C>PRINT A:LISTA.TXT
```

A parancs hatására a PRINT a memóriába töltődik, és a következő üzenetet kapjuk:

```
Name of List device [PRN]:
(a listázó eszköz neve [PRN]:)
```

Válaszként azt az output eszközt kell megadni, amelyikre a file-okat majd ki akarjuk iratni. Az < ENTER > megnyomása önmagában a PRN: használatát eredményezi.

2. Ha már elindult a nyomtatási folyamat, és a következőt adjuk meg:

```
A > PRINT A:LIST3.TXT/C A:LIST4.TXT A:LIST5.TXT
```

akkor a LIST3.TXT törlődik a várakozási sorból, a LIST4.TXT és a LIST5.TXT pedig bekerül a várakozási sorba.

3. Ha a teljes várakozási sort szeretnénk törölni, akkor a következőt kell megadni:

```
A > PRINT /T
```

A rendszer ekkor üzenetet is küld, hogy minden file-nevet törölt a várakozási sorból.

3.12 FASTOPEN - file-ok elérésének gyorsítása

A DOS, ha egy olyan file-t használunk, amelynek elérési útja hosszú, a file-műveletek során a teljes elérési utat végigjárja. Ez olyan programok esetén, amelyek gyakran fordulnak a lemezegységhez, sok időt elrabolhat.Ezt gyorsítja a FASTOPEN parancs.

Verzió: 3.30
A parancs típusa: külső
A parancs formája:

FASTOPEN *d*:[=*nnn*]...

A paraméterek jelentése:

d	annak a lemezegységnek az azonosítója, amelyiken a gyors elérést be akarjuk vezetni.
nnn	megadja azoknak a tartalomjegyzék és file-név bejegyzéseknek a számát, amelyek helyét a DOS a memóriában megjegyezhet. Ha ez a szám nagyobb, mint az általunk használt file-nevekben összesen szereplő tartalomjegyzék és file-nevek száma, akkor a file-ok elérése nagyon meggyorsul. nnn értéke legalább 10,legfeljebb 999 lehet. Alapértelmezése 34.

Példák:

```
FASTOPEN D:
FASTOPEN C: = 120 D: = 60
```

4. DOS: lemezkezelési parancsok

A DOS 2.00-ás verziója biztosította először a felhasználónak a többszintű tartalomjegyzékek létrehozását. Az ezt megelőző verziókban sem ez, sem a fix lemez kezelése még nem állt a felhasználók rendelkezésére. Az ebben a fejezetben ismertetésre kerülő parancsok a következő funkciókat látják el:

- többszintű tartalomjegyzék kezelése,
- tartalomjegyzék kijelzése,
- lemez formázása,
- lemez másolása,
- lemezek összehasonlítása,
- lemez ellenőrzése és
- írás utáni ellenőrzés beállítása.

4.1 MKDIR - altartalomjegyzék létrehozása

A parancs lehetővé teszi, hogy egy tartalomjegyzékben létrehozzunk egy altartalomjegyzék bejegyzést.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

```
MKDIR [d:]út
MD [d:]út
```

Megjegyzés:

1. Egy lemezen tetszőleges számú altartalomjegyzék előállítható, számukat csak a lemez kapacitása korlátozza. (A főtartalomjegyzékben azonban csak korlátozott számú - elsődleges - altartalomjegyzék bejegyzés lehet.)

2. Az út maximális hossza nem lehet nagyobb, mint 63 karakter, beleértve a \ (backslash) jeleket is.

3. Ha az út megadásánál \ (backslash) nem szerepel, akkor az altartalomjegyzék bejegyzés az aktuális tartalomjegyzékbe kerül.

4. Egy újonnan létrehozott tartalomjegyzék két bejegyzést tartalmaz. Ezek a . és a .. bejegyzések, amelyek a tartalomjegyzékre magára, illetve az ezt közvetlenül tartalmazó tartalomjegyzékre utalnak.

5. Egy tartalomjegyzéket akkor hívunk üresnek, ha a . és .. bejegyzéseken kívül nem tartalmaz mást. (Megjegyzés: Mivel altartalomjegyzékben lehetnek rejtett file-ok is, nem biztos, hogy ha a DIR parancs csak a . és .. bejegyzéseket mutatja, akkor az altartalomjegyzék üres.)

Példa:

1. Ha a főtartalomjegyzékben akarunk létrehozni egy altartalomjegyzék-bejegyzést (SZINT1), akkor a következőt adhatjuk meg:

```
MD \SZINT1
```

Ha további altartalomjegyzéket akarunk létrehozni (SZINT2), akkor a következőt adhatjuk meg:

```
MD \SZINT1\SZINT2
```

2. Ha az aktuális tartalomjegyzékbe akarjuk beírni a SZINT4 altartalomjegyzék bejegyzést, akkor a következőt adhatjuk meg:

```
MD SZINT4
```

4.2 CHDIR - aktuális tartalomjegyzék kijelölése

A parancs segítségével egy megadott tartalomjegyzék aktuális tartalomjegyzékként kijelölhető.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

CHDIR [[*d*:]*út*]
CD [[*d*:]*út*]

Megjegyzés:

1. A DOS minden file-t, amit út kijelölés nélkül adunk meg, az aktuális tartalomjegyzékben keres.
2. Ha csak a CD vagy a CHDIR kulcsszót adjuk meg, akkor az aktuális tartalomjegyzék elérési útját írja ki válaszul a rendszer.

Példa:

1. A 2.5 szakasz elején adott ábra szerinti struktúrában a \PROG1-be a következő úton juthatunk:

```
CD\EGYETEM\KAR1\PROG1
```

A parancs kiadása után az aktuális tartalomjegyzék a PROG1 lesz.

2. Ha a következőt adjuk meg:

```
CHDIR \
```

akkor az aktuális meghajtó főtartalomjegyzéke lesz az aktuális tartalomjegyzék.

3. Ha az aktuális meghajtón levő aktuális tartalomjegyzékre vagyunk kíváncsiak, akkor a

```
CD
```

parancs kiadásával kaphatjuk ezt meg. A

```
CD D:
```

parancs kiadására válaszul a D:egység aktuális tartalomjegyzékét írja ki a DOS.

4.3 RMDIR - altartalomjegyzék törlése

A parancs lehetővé teszi, hogy a megadott tartalomjegyzékből altartalomjegyzéket töröljünk.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

RMDIR [*d:*]*út*
RD [*d:*]*út*

Megjegyzés:

1. Egy altartalomjegyzék csak akkor törölhető, ha az sem altartalomjegyzék, sem file bejegyzést már nem tartalmaz, ezért az altartalomjegyzékek törlését mindig a legalsó szinten kell kezdeni.
2. A főtartalomjegyzék és az aktuális tartalomjegyzék nem törölhető.
3. Az út megadásában szereplő utolsó tartalomjegyzéknév a törlendő tartalomjegyzék.
4. Ha egy látszólag üres altartalomjegyzéket a parancs nem hajlandó törölni, akkor vagy ez az aktuális (akkor át kell térni másra), vagy rejtett file-okat tartalmaz (melyeket a DIR nem jelez ki). Utóbbi esetben csak speciális programokkal nézhetjük meg, milyen file-ok vannak itt és csak ilyenekkel tudjuk ezeket törölhetővé tenni.

Példa:
1. Ha a következőt adjuk be:

```
RD \SZINT1\SZINT2\SZINT3
```

akkor a SZINT2 tartalomjegyzékből a SZINT3 törlődik.

2. Ha a teljes fastruktúrát törölni szeretnénk, akkor azt a következő lépésekben tehetjük:

```
RD \SZINT1\SZINT2
RD \SZINT1
```

Természetesen előtte az altartalomjegyzékből a file-okat is törölni kell.

4.4 TREE - tartalomjegyzék struktúrájának kiíratása

A parancs lehetővé teszi, hogy egy lemez összes elérési útját kilistázzuk.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

```
TREE [d:][/F]
```

Megjegyzés:

1. Minden megtalált tartalomjegyzéknél a teljes elérési utat kijelzi és megadja az altartalomjegyzékeinek a listáját is.
2. Ha a /F paramétert megadjuk, akkor az altartalomjegyzékben szereplő valamennyi file-nak az azonosítóját is kijelzi.
3. Az outputot célszerű a nyomtatón is kiíratni, vagy átirányítani. (Lásd 12. fejezet "A standard I/O átirányítása" című alfejezetet.)
4. A PROPOS-16 operációs rendszerben az FSYS parancs végzi el a TREE funkciót. Paraméterezése is azonos.

Példa:

1. A nyomtatóra átirányítva a kiíratást:

```
TREE C:/F > PRN
```

2. A lista egy file-ba kerül, amelyet a PRINT paranccsal kiírathatunk. Ekkor a következőt adjuk meg:

```
TREE C:/F > TARTJEGY.LST
```

4.5 PATH - bejárási utak kijelölése programhoz

A parancs lehetővé teszi olyan programok elérését is, amelyek az aktuális tartalomjegyzéken kívül találhatók. Segítségével több elérési út is kijelölhető, illetve a kijelölt utak lekérdezhetők.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

```
PATH [[d:]út[[;[d:]út]...]]
```

vagy

```
PATH;
```

Megjegyzés:

1. Az útkijelöléseket mindig pontosvessző (;) választja el. Ha a DOS a file-t az aktuális tartalomjegyzékben nem találja, akkor a keresést abban a sorrendben végzi, ahogyan azt a felhasználó a parancsban meghatározta.

   Abban az esetben, ha több azonos nevű file is van az útkijelölések tartalomjegyzékeiben, akkor azt fogja megtalálni, amelyiket a parancs megadásakor előbb jelöltünk ki.

2. Az útkijelölések a következő paranccsal törölhetők:

   PATH ; (a parancsot csak egy pontosvessző követi)

   Ha a rendszert újraindítjuk, a DOS már "nem emlékszik" az útkijelölésekre. (Kivéve arra, ami az AUTOEXEC.BAT-ben szerepel.)

3. Ha az érvényben lévő útkijelöléseket akarjuk kiíratni, akkor a következőt kell beírni:

   PATH

4. Az útkijelölésben lévő helytelen információkat a DOS csak akkor fedezi fel, ha valamely bejárási utat végignézi. A hibás, nem létező utakat átlépi, és a következő útkijelölésnél folytatja a keresést.

5. A PATH parancs segítségével csak a .COM, .EXE és .BAT kiterjesztésű file-okat érhetjük el. Ha ettől eltérő kiterjesztésű file-jaink vannak, akkor azok elérése az APPEND (3.30 és későbbi verziók) parancsot használjuk.

Példa:

1. Egy út kijelölése a következő lehet:

   PATH C:\SZINT1\SZINT2\SZINT3

2. Több út megadására példa az alábbi:

   PATH UTVONAL; EGYETEM; EGYETEM KAR1

4.6 FORMAT - lemez formázása

A parancs lehetővé teszi, hogy a lemezt előkészítsük DOS file-ok befogadására. A parancs ellenőrzi, hogy a lemez nem sérült-e. Felírja az üres tartalomjegyzék-blokkot és inicializálja a file-leíró táblázatot (FAT).

Verzió: 2.11 - 3.30
A parancs típusa: Külső

A parancs formája:

FORMAT [*d*:][/S][/1][/8][/4][/V][/B][/N:*xx*][/T:*yy*]

A paraméterek jelentése:

d:	azt a meghajtót adja meg, amelyen a formázandó lemez van.
/S	az aktuális meghajtón lévő lemezről az operációs rendszer file-jait az új, most formázandó lemezre másolja. Ha e paramétert használjuk, és az aktuális meghajtón éppen nincs rajta a rendszer, két eset van. Ha az aktuális meghajtó a Winchester, akkor a rendszer felszólít, hogy tegyük az "A" meghajtóba a rendszerlemezt. Ha az aktuális meghajtón cserélhető a lemez, akkor oda kéri a rendszerlemezt.
/1	azt jelöli, hogy a lemezt egyoldalas felhasználásra formázzuk.
/8	azt jelöli, hogy a lemezt sávonként 8 szektorral formázza. Alapértelmezés: 9 szektor/sáv.
/4	használata biztosítja, hogy a nagy-kapacitású lemezegységben (1.2 Mbyte-os) a lemezt normál (1 vagy 2 oldalas) lemezként formázza meg a parancs. Nem használható a /1, illetve /8 paraméterekkel.
/V	lehetővé teszi, hogy kötetazonosítót adjunk a lemeznek. Ez a lemez azonosítására szolgál. (PROPOS-16 esetén /L a paraméter.)
/B	Egy olyan lemezt, amelyet sávonként 8 szektorosra formázunk, úgy formattál, hogy legyen rajta hely az IBMBIO.COM és az IBMDOS.COM rendszerprogramoknak. A /B paraméter sem ezeket, sem a parancsfeldolgozó rendszerprogramot (command processor) nem teszi rá a lemezre, csak helyet biztosít arra, hogy majd a DOS bármely változatát erre a lemezre a SYS paranccsal rátehessük.
/N:*xx*	megadja az egy sávra írandó szektorok számát.
/T:*yy*	megadja a megformázandó sávok számát.

Megjegyzés:

1. A /1, a /4 és a /8 paraméter fix lemez esetén nem használható.
2. A /V és a /S paramétereket nem lehet a /B paraméterrel együtt használni.
3. A /S esetén a következő file-ok másolódnak a lemezre:

   IBMBIO.COM
   IBMDOS.COM
   COMMAND.COM (parancsfeldolgozó rendszerprogram).

4. Az alábbi táblázat összefoglalja, hogy adott típusú lemezegység esetén ilyen paraméter használható:

Típus	**Megengedett paraméter**						
fix lemez	/S	/V					
1.2 Mbyte	/S	/V	/4				
720 Kb/144 Mb	/S	/V	/N	/T			
320/360 Kbyte	/S	/V	/1	/8	/B		
160/180 Kbyte	/S	/V	/1	/8	/B	/N	/T

5. A formázás során hibásnak talált lemezterületeket a program fenntartott sávoknak jelöli meg, így megakadályozza, hogy ezekre a területekre file-ok kerüljenek.

6. Ha a /V paramétert megadtuk, akkor a parancs bekéri a kötetazonosítót, ez maximálisan 11 karakter lehet. Azok a karakterek használhatók, amelyek a file-névben is előfordulhatnak.

7. A fix lemezeket többnyire már a gyártók formázzák. A formázás során a DOS partíció minden szektora ellenőrzésre kerül, de újraformázás nem történik, így gyakran előfordul, hogy néhány kbyte hibás, ettől ne ijedjünk meg.

8. Azok a floppy lemezek, amelyeket a gyártó egyoldalasként jelölt meg, többnyire megformázhatók kétoldalasra is.
   Az egyoldalas lemezek egy- és kétoldalas meghajtón is használhatók (de célszerűen egyoldalasra formázva).
   A kétoldalas lemezek egyoldalas meghajtón is használhatók (természetesen csak egyik oldalukat használhatjuk!).

9. A FORMAT a következő üzeneteket küldi:

   - a lemez teljes tárolókapacitása,
   - a hibás terület mérete,
   - a rendszer file-oknak fenntartott terület,
   - a rendelkezésre álló szabad terület nagysága.

10. A DOS 3.20 és későbbi verziók a fix lemez véletlen megformázása ellen bizonyos védelmet tartalmaznak, nevezetesen a DOS megkérdezi a fix lemez (jelenlegi) kötetazonosítóját. Ha ezt nem tudjuk megadni, a formázás elmarad. Ha nem tudjuk a kötetazonosítót, akkor a FORMAT parancs kiadása előtt adjuk ki a VOL parancsot!

11. A parancs ismételt beírása nélkül több lemez azonos módon megformázható.

12. Az /N és a /T paramétereket egyszerre kell használnunk.

13. A parancs a következő kódokkal tér vissza:

Kód	Jelentés
0	Sikeres befejezés (az utolsó formázásé)
1-2	Nem használt
3	< Ctrl-Break > -kel megállítva
4	Hiba történt
5	Fix lemez formázásakor N-nel válaszoltunk.

14. A FORMAT parancsot ne adjuk ki olyan meghajtóra, amire a SUBST vagy JOIN parancsokban hivatkozunk. A FORMAT hálózati lemezegységre sem adható ki.

Példa:

Ha a konfigurációnk egyetlen floppy meghajtót tartalmaz, akkor a lemez formázása a következő módon történhet:

```
A>FORMAT A:
```

Ezután a rendszer a következő üzenetet küldi:

Insert new diskette for drive A:
and strike any key when ready
(Tegye az új lemezt az A: meghajtóba,
majd nyomjon meg egy billentyűt.) (DOS 2.xx)

illetve

Insert new diskette for drive A:
and press ENTER when ready
(Tegye az új lemezt az A: meghajtóba,
majd nyomja meg az ENTER billentyűt.)

Ezt követően a rendszer elkezdi a lemez formázását, és a következő üzenetet küldi:

Formatting ...
(Formázás folyik)

A DOS 3.20 és későbbi verziói egy Head:xxx Cylinder:yyy üzenetet is küldenek, amely segítségével nyomon követhető a formázás előrehaladása.

A formázás befejezésekor a következő üzenetet küldi:

Formatting... Format complete
362496 bytes total disk space
362496 bytes available on disk
(Formázást befejezte.
362496 byte a teljes lemezkapacitás.
362496 byte a szabad terület a lemezen.)

Ezután a következő üzenetet küldi:

Format another (Y/N)?
(Kell-e még másik lemezt formázni? (I/N)?

Erre az Y választ adjuk, ha igen, egyébként N-et.

Fix lemez formázásakor a program megkérdezi, valóban formázni akarjuk-e Winchesterünket:

WARNING! ALL DATA ON NON-REMOVABLE DISK DRIVE d: WILL BE LOST!
Proceed with Format (Y/N)?

(Figyelem! A fix lemezen levő összes adatállomány el fog veszni!
Valóban el akarja indítani a formázást?)

Amennyiben Y-t válaszolunk, a fix lemez formázása elkezdődik. Ettől a pillanattól kezdve hiába állítjuk le vagy kapcsoljuk ki a gépet, a file-ok már elvesztek!

4.7 SELECT - új rendszerlemez előállítás adott billentyűzet rutinnal, dátum és idő formátummal

A parancs lehetővé teszi, hogy egy új DOS lemezt állítsunk elő megadott billentyűzet rutinnal, dátum és idő formátummal.

Verzió: 3.00 - 3.30

A parancs típusa: Külső

A parancs formája:

SELECT [[A: | B:] *d*:[*út*]] *xxx yy*

A paraméterek jelentése:

A: vagy B: a forrásmeghajtót jelenti. Csak ezeket az egységeket használhatjuk. Ha egyiket sem adjuk meg, akkor az A: egységet használja a parancs. A d:[út] a cél tartalomjegyzéket adja meg, ide másolja a DOS a parancs file-okat. Ha nem adjuk meg, akkor a főtartalomjegyzékbe kerülnek. xxx az ország kódját adja meg. yy a billentyűzet rutin kódját adja. Lehetséges értékeik a következők:

Ország	**Kód**	**Billentyűzet rutin**
Ausztrália	061	•
Belgium	032	•
Kanadai francia	002	•
Dánia	045	•
Finnország	358	•
Franciaország	033	FR
NSZK	049	GR
Olaszország	039	IT
Izrael	972	•
Közel Kelet	785	•
Hollandia	031	•
Norvégia	047	•
Portugália	351	•
Spanyolország	034	•
Svédország	046	•
Svájc	041	•
Egyesült Királyság	044	UK
Egyesült Államok	001	US

A •-gal jelölt országokban nincs külön billentyűzet rutin. Itt tetszőleges yy megadható és egy KEYByy.COM nevű file-nak kell tartalmaznia a billentyűzet rutint.

Megjegyzés:

1. A SELECT parancs a DOS 3.00-nál korábbi verzióiban nem használható.
2. A SELECT parancs végrehajt egy FORMAT parancsot is, tehát a cél-lemezen mindent töröl!
3. A SELECT a cél-lemezen létrehoz egy CONFIG.SYS file-t egyetlen COUNTRY = xxx paranccsal. Ezenkívül a SELECT generál még egy AUTOEXEC.BAT file-t is az alábbi utasításokkal:

```
PATH ;[,út;]
KEYB: yy xxx
DATE
TIME
VER.
```

4.8 SYS - rendszer-file-ok másolása

A parancs lehetővé teszi, hogy az operációs rendszer a megadott meghajtóra kerüljön.

Verzió: 2.11 - 3.30
Parancs típusa: Külső
Parancs formája:

SYS *d*:

Megjegyzés:

1. Az átmásolás a következő esetekben lehetséges:
   - a lemez formázott, de a tartalomjegyzéke teljesen üres, vagy
   - a lemezt a /B vagy a /S paraméter felhasználásával már formáztuk.
2. Ahhoz, hogy a lemezt rendszerlemezként használjuk, a következőknek kell teljesülni:
   - Az IBMBIO.COM és az IBMDOS.COM file-oknak a tartalomjegyzékben az első két bejegyzésnek kell lenni.
   - Az IBMBIO.COM-nak a legelső két egymást követő adatszektorban kell elhelyezkednie.
3. A PROPOS-16 operációs rendszerben ezt a parancsot nem valósították meg.
4. Néhány alkalmazói rendszert szállítanak úgy, hogy a lemezen már kijelölték a rendszer file-jait, de azok még nem kerültek fel a lemezre. A felhasználó az általa használt DOS verziót a SYS paranccsal helyezheti el a lemezen, amit ezután már rendszerlemezként is használhat.
5. A SYS parancs kiadása után még át kell másolni a cél-lemezre a COMMAND.COM-ot is.

4.9 DIR - tartalomjegyzék kiíratása

A parancs listázza vagy a lemezen lévő összes file-ra, vagy megadott file-ra, illetve file-okra vonatkozó tartalomjegyzék bejegyzéseket. A parancs a következő információkat minden esetben szolgáltatja:

- a lemez azonosítóját (legfeljebb 11 karakter),
- a rendelkezésre álló szabad területet,
- a file-ra vonatkozó információkat:
  - azonosítóját,
  - méretét (decimális számként, byte-okban),
  - az utolsó módosítás dátumát, időpontját.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

DIR [*d*:][*út*][*file-név*[.*kit*]][/**P**][/**W**]

A paraméterek jelentése:

/P paraméter biztosítja, hogy az információ 24 soronként íródik ki a képernyőre. Egy tetszőleges billentyű lenyomásával folytatódik a kijelzés.

/W paraméter egy rövidített, tömör kijelzést tesz lehetővé. A file azonosítója kerül csak kijelzésre. A 80 oszlopos képernyőre történő kiíratásnál egy sorba öt file-azonosító kerül.

Megjegyzés:

1. A rejtett (hidden) file-oknál nem jelenik meg információ.
2. A tartalomjegyzékben azok a bejegyzések, amelyek altartalomjegyzékre vonatkoznak, < DIR > megjelöléssel kerülnek kijelzésre.
3. A file-ok megadásánál a globális név-jelek is használhatók: a csillag (*) és a kérdőjel (?).
4. A dátum és idő formája a COUNTRY-ban megadott értéknek megfelelően jelenik meg.

A parancs két formában alkalmazható:

I. Alkalmazás - a megadott tartalomjegyzék valamennyi file-jának kijelzése

A parancs formája:

DIR [d:][út]

Ha sem meghajtót, sem útkijelölést nem adunk meg, akkor az aktuális meghajtón lévő lemez aktuális tartalomjegyzékét írja ki.

Az út kijelölése lehetővé teszi egy altartalomjegyzék kiíratását is.

II. **Alkalmazás** - egy tartalomjegyzék meghatározott elemeiről kapunk információt

A parancs formája:

DIR [d:][file-név[.kit]]

Ha csak a file-nevet adjuk meg, akkor a file-név kiterjesztésénél a csillag (*) a feltételezett. Ha olyan file-t akarunk kiíratni, amelynek nincs kiterjesztése, a neve után a pontot (.) meg kell adni.

Megjegyzés:

1. A tartalomjegyzék kijelzésénél két speciális bejegyzés is előfordulhat. Az egy pont (.) az éppen kijelzés alatt álló altartalomjegyzékre vonatkozik. A két pont mindig az aktuális tartalomjegyzék feletti tartalomjegyzéket jelöli. Így például megengedett a következő parancs:

DIR ..

A tartalomjegyzék kijelzésénél a \ (backslash) a főtartalomjegyzéket jelöli.

Példa:

Amennyiben az altartalomjegyzékeket úgy különböztetjük meg a normál file-októl, hogy nem adunk kiterjesztést, akkor a

DIR *.

parancs ezeket fogja kiírni (azaz pontosan azokat a file-okat, melyeknek nincs kiterjesztése).

4.10 VOL - kötetazonosító kiíratása

A parancs lehetővé teszi, hogy a megadott meghajtón lévő lemez kötetazonosítóját kiírassuk.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

VOL [*d*:]

Megjegyzés:

1. Ha meghajtót nem adunk meg, akkor az aktuális meghajtón lévő lemez azonosítóját jelzi ki.
2. Kötetazonosítót a lemeznek formázáskor vagy LABEL paranccsal adhatunk. A kötetazonosító maximálisan 11 karakter hosszúságú lehet.

3. Ha nincs kötetazonosítója a lemeznek, akkor a következő üzenetet kaphatjuk:

```
Volume in drive A has no label
```

(Az A: meghajtón lévő lemeznek nincs azonosítója.)

4. A DIR parancs az első kiírt sorában szintén kiírja a kötetazonosítót.

4.11 LABEL - kötetazonosító megadása

A parancs segítségével módosíthatjuk egy lemez kötetazonosítóját.

Verzió: 3.00 - 3.30
A parancs típusa: Külső
A parancs formája:

LABEL [*d*:][*cimke*]

Megjegyzés:

1. A *cimke* 1-11 karakterből állhat.

2. Ha nem adjuk meg a cimké-t a parancsban, akkor a DOS kiírja a jelenlegi kötetazonosítót és utána megkérdezi az új kötetazonosítót. Ha erre az ENTER azonnali megnyomásával válaszolunk, akkor újabb kérdés következik:

```
Delete current volume label (Y/N)?
```

(Töröljem a kötetazonosítót (I/N)?) Az Y válaszra a kötetazonosító törlődik, az N-re megmarad az eredeti.

3. A LABEL-t nem szabad a SUBST-tal és az ASSIGN-nal együtt használni.

4. A LABEL hálózaton implementált lemezegységre nem használható.

5. A LABEL a DOS 3.00-nál régebbi verzióiban nem használható.

4.12 DISKCOPY - lemez másolása

A parancs lehetővé teszi, hogy egy floppy lemez tartalmát átmásoljuk egy másik floppy lemezre.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

DISKCOPY [*d*: [*d*:]][/1]

A paraméterek jelentése:

Az első meghajtó a forrás-, a második a célmeghajtót jelöli. A /1 paraméter lehetővé teszi, hogy a lemeznek csak az egyik oldalát másoljuk át.

Megjegyzés:

1. A parancs egy-meghajtós rendszerben is használható. Ekkor a program üzenetet küld, és így kéri a felhasználót a lemez cseréjére. Egy billentyű lenyomása után folytatja a működését.
2. Ha számítógépünk lemezegysége egyoldalas lemezhez való, akkor egyoldalas lemezeket másolhatunk egyoldalasokra. Ha kétoldalashoz való, másolhatunk egyoldalast egyoldalasra, és kétoldalast kétoldalasra.
3. A paranccsal több lemez is átmásolható. A másolás befejezése után a program megkérdezi, kér-e a felhasználó további másolást.
   Ha a válasz Y (Igen), akkor a meghajtó kijelölések továbbra is érvényben maradnak. Ha a válasz N (Nem), akkor a parancs befejezi működését.
4. Ha a cél-lemez még nem formázott, vagy nem egyezik a típusa a forrás lemezével, akkor a parancs a lemez formázását is elvégzi.
5. A parancs szektoronként másolja át az információt. A file-ok a céllemezen ugyanabba a szektorba kerülnek, ahol a forrás-lemezen voltak. A lemez átmásolása előtt célszerű a lemez sűrítését is elvégezni. Ez a

   ```
   COPY d:*.*
   ```

   segítségével történhet, csak ezután célszerű a lemez másolását végrehajtani. A másolás és a sűrítés egy lépésben is végrehajtható, ha a következő parancsot használjuk:

   ```
   COPY [d:]*.* d:
   ```
6. Ha valamelyik lemezen hiba fordul elő, akkor a DISKCOPY üzenetet küld. Ebben leírja a hiba okát és kijelzi a hibás területet: meghajtó, oldal és sáv. Ezután a program folytatja a másolást. A cél-lemez használhatósága attól függ, hogy a hiba file-t tartalmazó adatterületen fordult-e elő.
7. Ha a DISKCOPY befejezte működését, a két lemez összehasonlítását is célszerű elvégezni. Ezt a DISKCOMP paranccsal végezhetjük el.
8. A DISKCOPY nem használható az ASSIGN-nal és a SUBST-tal együtt.
9. A DISKCOPY nem használható hálózaton implementált egységre. Ilyenkor célszerű helyette a COPY *.* használata.
10. A PROPOS-16-ban járulékos [/C[0]] paraméterek is megadhatók. C paraméter esetén a másolás után ellenőrzés történik, hiba esetén hibajelzéssel. Ha az 0 paramétert is megadjuk, csak összehasonlítás történik, másolás nem.

Példa:

Ha az A: meghajtón lévő lemezt a B: meghajtóban lévő lemezre szeretnénk másolni, akkor a következőképpen célszerű elvégezni:

1. A forrás lemez file-jainak a tömörítése (átmásolással lehetséges):

```
C>COPY A:*.* B:/V
```

2. A lemez visszamásolása:

```
C>DISKCOPY B: A:
```

3. A lemezek összehasonlítása:

```
C>DISKCOMP A: B:
```

4.13 DISKCOMP - lemezek összehasonlítása

A parancs lehetővé teszi, hogy a DISKCOPY paranccsal történő lemezmásolás után a cél- és a forrás-lemez tartalmát összehasonlítsuk. A paranccsal teljes lemezek összehasonlítása végezhető, míg a COMP csak file-okat hasonlít össze.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

DISKCOMP [*d*:[*d*:]][/1][/8]

A paraméterek jelentése:

A *d*-k a floppy meghajtók azonosítói

/1 paraméter jelöli, hogy csak egyik oldalát kell a lemezeknek összehasonlítani, akkor is, ha azok kétoldalasak.

/8 paraméter esetén a lemezeknek sávonként csak nyolc szektora kerül összehasonlításra, akkor is, ha azok 9 szektor/sáv típusúak.

Megjegyzés:

1. A parancs fix lemez esetén nem használható. Ha a fix lemez azonosítóját adjuk meg, hibaüzenetet kapunk.

2. A parancs egy-meghajtós rendszerben is használható. Ekkor a program üzenettel kéri fel a felhasználót a lemezek cseréjére. Egy tetszőleges billentyű lenyomása után folytatja az ellenőrzést.

3. Ha a DISKCOMP befejezte a két lemez összehasonlítását, akkor üzenetet küld. Megkérdi a felhasználót, hogy akarja-e még az összehasonlítást folytatni. Ha igen, akkor a válasz Y. Az összehasonlítás ugyanazokon a meghajtókon történik. Ha nem, akkor a válasz N.

4. A program sávonként végzi az összehasonlítást. Üzenetet küld, ha ezek nem egyeznek. Ebben megadja az eltérő sáv sorszámát (0-39.), és azt, hogy melyik oldalon található (0-1.).

5. Ha nem adunk meg meghajtó azonosítót, akkor az összehasonlítás az aktuális meghajtón történik, mintha csak egy meghajtó lenne a rendszerben.

6. Ha nem adunk meg paramétert, akkor a lemez típusának megfelelően végzi az összehasonlítást:
    - Mindkét oldal dupla oldalú lemezként.
    - 9 szektor/sáv.
7. Ha a COPY paranccsal másoltunk egy lemezt, akkor a DISKCOMP paranccsal történő összehasonlítás hibát jelez, mivel a COPY a file-okat általában nem ugyanarra a helyre másolja a cél-lemezen, mint ahol az a forrás lemezen volt.

    A DISKCOMP az összehasonlítást oldalanként, sávonként, szektoronként, byte-ról byte-ra végzi.
8. A parancs a hiba kijelzése után mindig folytatja az összehasonlítást, függetlenül attól, hogy eddig hány eltérést jelzett. (A COMP tíz eltérés után befejezi a működését.)
9. A DISKCOMP parancs nem használható a SUBST vagy a JOIN paranccsal együtt.
10. A DISKCOMP nem tudja értelmezni az ASSIGN-ban megadott értékeket.
11. A DISKCOMP hálózaton implementált lemezegységre nem használható.

Példa:

1. Ha csak a következőt adjuk meg:

```
A > DISKCOMP
```

akkor az A: meghajtón történik az összehasonlítás. A meghajtóba először a forrás lemezt kell betenni, majd a rendszer üzenetet küld és felkér a lemezek cseréjére. Ha kész, le kell nyomni egy billentyűt, és az öszehasonlítást folytatja.

Ha befejezte az összehasonlítást, akkor megkérdezi, kell-e további összehasonlítást végeznie. Az Y, illetve a J válasz esetén folytatja az összehasonlítást, N válasz esetén a program befejezi működését.

4.14 CHKDSK - lemez vizsgálata

A parancs információt ad a lemez állapotáról a következő paraméterekkel:

- kötetazonosító,
- a lemez teljes kapacitása,
- védett file-ok mérete,
- felhasználói file-ok mérete,
- a szabad lemezterület nagysága,
- a RAM memória mérete,
- a szabad RAM memória mérete.

Helyreállítja a file-okat, ha ellentmondás van a file leíró táblázat (file allocation table - FAT) és a tárolt adatterületek között. A parancs hálózaton implementált lemezegységre nem adható ki!

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

CHKDSK [*d*:][*file-név*[.*kit*]][/F][/V]

A paraméterek jelentése:

Ha egy file nevét adjuk meg, akkor a file nem összefüggő területeinek a számát adja meg.

/F paraméter megadásakor a program felújítja a FAT-ot. A FAT javítását a fizikai tárolásnak megfelelően végzi el. Ha nem adjuk meg a /F paramétert, akkor hibás FAT esetén csak rövid hibaüzenetet küld és kijelzi a lehetséges javítást, de azt nem végzi el.

/V paraméter megadása esetén a CHKDSK részletes üzenetet küld.

Megjegyzés:

1. A CHKDSK csak a fő tartalomjegyzéket használja.
2. Ellenőrzés után a parancs a lemez állapotáról információkat is küld.
3. A program a védett file-okról is ad információt.
4. A CHKDSK nem vár arra, hogy feltegyük a lemezt, ezért egy cserélhető lemezegységet tartalmazó rendszer esetén meg kell adni a kívánt meghajtót. Ha nem adjuk meg, akkor az aktuális meghajtón lévő lemezt ellenőrzi.
5. A parancs fix lemez és floppy esetében is használható.
6. Minden javítást kijelez a CHKDSK, az Y és az N beadása után az üzenet végét jelző billentyűt (< ENTER >) is le kell nyomni.
7. A globális név-jelek használhatók. Ha a következő parancsot adjuk meg:

   CHKDSK [d:]*.*

   akkor a lemezen a file-ok tördeltségéről kapunk információt.
8. Ha a CHKDSK különálló adatterületeket talál, akkor rákérdez, hogy a file-ok különálló területeit összegyűjtse-e.

   Ha a /F paramétert megadjuk és a válaszunk Y, akkor a területeket a következő azonosítóval gyűjti össze:

   FILEnnnn.CHK

   ahol nnnn a file sorszáma.

   A file-ok azonosítóját a tartalomjegyzékbe írja. A felhasználó ezután ellenőrizheti, hogy számára hasznos információt tartalmaznak-e. Ha nem, akkor törölheti ezeket.
9. A PROPOS-16 rendszerben a CHKDSK funkcióját a STAT parancs látja el.

4.15 VERIFY - írás utáni ellenőrzés

A parancs a lemezre történő írás ellenőrzését biztosítja.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

```
VERIFY [ON | OFF]
```

A parancs működése:

A VERIFY ON kiadása után a rendszer minden lemezes írásművelet után ellenőrzi, hogy az éppen felírt információ visszaolvasható-e. Ez jelentősen megnöveli az I/O műveleti időket. Használatát csak indokolt esetben ajánljuk. A VERIFY OFF kiadásával az írás-ellenőrzést kikapcsoljuk. Ha csak a VERIFY kulcsszót adjuk be, akkor a kapcsoló aktuális értékét jelzi ki a DOS (ON-OFF).

Megjegyzés:

Néhány parancs működése alatt is megadható az írásellenőrzés lehetősége. Például a COPY /V paramétere.

Példa:

Ha a VERIFY kapcsoló értékét akarjuk lekérdezni, akkor az a következőképpen történhet:

```
VERIFY
```

VERIFY =OFF, ha a kapcsoló nem volt, és VERIFY =ON ha a kapcsoló be volt kapcsolva.

4.16 SUBST - tartalomjegyzék helyettesítése lemezegység betűjelével

A parancs lehetőséget biztosít arra, hogy egy adott tartalomjegyzékre, mint új lemezegységre hivatkozhassunk. Ez főleg olyan felhasználói programok esetén jó, amelyik nem ismeri a tartalomjegyzékeket.

Verzió: 3.10 - 3.30
Parancs típusa: Külső
Parancs formája:

SUBST [*d: d1:út*] | [*d:* **/D**]

A továbbiakban a *d:* lemezegység a *d1:út* által kijelölt tartalomjegyzékkel lesz azonos. A *d:***/D** alak használata megszünteti a kijelölés hatását.

Példa:

```
SUBST E: C:\WORK
```

A fenti parancs kiadása után az E: lemezegység logikailag azonos lesz a C:\WORK tartalomjegyzékkel. A SUBST E:/D visszaállítja az eredeti állapotot.

Megjegyzés:

1. Ha a SUBST parancsot többször is használjuk, akkor a CONFIG.SYS file-ban el kell helyezni egy megfelelő LASTDRIVE = parancsot is.
2. A helyettesített lemezre ne használjuk a CD, MD, RD és PATH parancsokat.
3. SUBST használatakor nem használhatjuk az alábbi parancsokat: ASSIGN, BACKUP, DISKCOMP, DISKCOPY, FDISK, FORMAT, JOIN, LABEL, RESTORE.
4. A paraméter nélküli SUBST kiírja az érvényben levő helyettesítéseket. A fenti példa esetén:

```
E: = > C:\WORK
```

4.17 JOIN - lemezegység és tartalomjegyzék összekapcsolása

A parancs segítségével egy lemezegységet egy tartalomjegyzékkel logikailag összekapcsolhatunk.

Verzió: 3.10 - 3.30
A parancs típusa: Külső
A parancs formája:

JOIN [*d1: d2:\tartalomjegyzék*] | [*d*:/D]

A paraméterek jelentése:

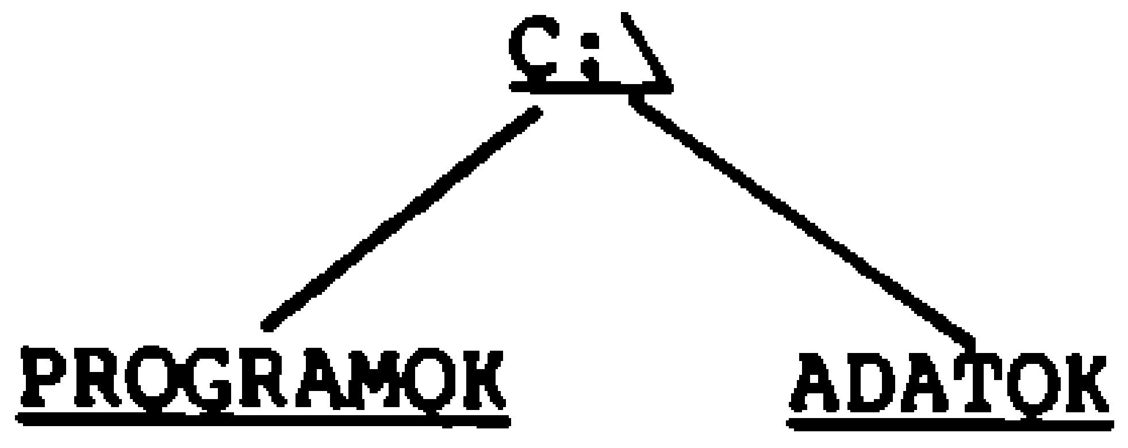
A *d1:* lemezegységet kapcsolja össze (azonosítja) a JOIN parancs a *d2:* lemezegység *tartalomjegyzékével*. Mint a parancs alakjából is látható, ez a tartalomjegyzék közvetlenül a főtartalomjegyzékben kell, hogy elhelyezkedjen.

A *tartalomjegyzék* a *d2:* egységen vagy üres kell, hogy legyen, vagy egyáltalán nem létezik.

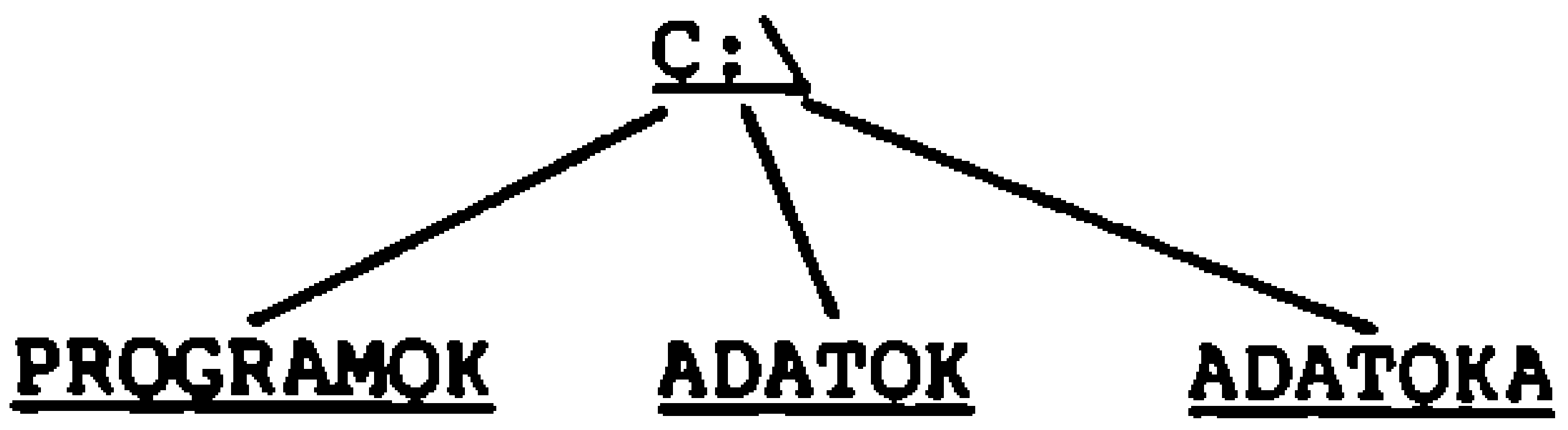
A /D paraméter törli egy előzőleg létrehozott JOIN hatását.

Megjegyzés:

1. A JOIN segítségével az adott lemezhez egy üres altartalomjegyzéket kapcsolunk hozzá, amelyik a másik lemezegységet reprezentálja. Tegyük fel, hogy a C: lemezegység altartalomjegyzékei a következőképpen helyezkednek el:



A JOIN A: C:\ADATOKA parancs kiadása után az a struktúra így alakul:



2. Az aktuális lemezegység nem kapcsolható össze egyetlen altartalomjegyzékkel sem. Ha megkíséreljük, akkor INVALID PARAMETER hibaüzenetet kapunk.
3. Kapcsolt lemezegységre nem lehet hivatkozni. Ha az 1. megjegyzésben szereplő szituációban kiadjuk a DIR A: parancsot, akkor INVALID DRIVE SPECIFICATION hibaüzenetet kapunk. Helyette a CD ADATOKA, majd a DIR parancsot kell kiadni.
4. A JOIN hálózaton implementált lemezegységre nem használható.
5. A SUBST, illetve ASSIGN használatával együtt a JOIN előre nem látható hibákat eredményezhet.
6. A BACKUP, RESTORE, FORMAT, DISKCOPY és a DISKCOMP parancsokat ne használjuk addig, míg a JOIN érvényben van!
7. A paraméter nélküli JOIN kiírja az adott összekapcsolásokat. Az 1-beli szituációban a JOIN kiadása után az alábbi üzenetet kapjuk:

   A: = > C:\ADATOKA
8. A JOIN a DOS 3.1-nél régebbi változataiban nem használható.

4.18 APPEND - bejárási utak kijelölése tetszőleges file-hoz

Az APPEND parancs lehetővé teszi olyan file-ok elérését is, amelyek az aktuális tartalomjegyzéken kívül vannak, anélkül, hogy az eljárási utat mindig újból-és újból leírnánk. Egyidőben több elérési út is kijelölhető. A paranccsal egyben le is kérdezhetjük a kijelölt utakat.

Verzió:3.30
A parancs típusa: Külső, majd rezidens
A parancs formája:

APPEND *d:út*[;[*d:*]*út*...]
APPEND [/X][/E]
APPEND [;]

A paraméterek jelentése:

d:út annak az elérési útnak amegadása, amit a DOS-nak - az aktuális mellett - végig kell néznie.

/X a kapcsoló utasítja a programot arra, hogy a DOS 11H, 4EH és 4BH funkcióinál megadott file-okat is keresse a benne megadott utakon. Különben ezt csak a 0FH, 23H és 3DH funkciókra vonatkozik.

/E a kapcsoló megadásakor az APPEND-ben megadott utakat a programkörnyezetben (environment) helyezi el a parancs.

Megjegyzések:

1. Az APPEND első hívásakor csak az 1. és 2. formát használhatjuk. Ilyenkor a parancs előtt úttal is megadhatjuk, hogy melyik tartalomjegyzékben található. Ezt követően az APPEND a memóriában marad, a DOS rezidens részévé válik. Ezért az APPEND előtt már nem szerepelhet útkijelölés.
2. Ha már egyszer a memóriába töltöttük az APPEND programot, akkor a parancs 1. és 3. formáját használhatjuk.
3. Mindig az utoljára használt APPEND-ben levő út-megadások érvényesek. Ezért az APPEND; törli az összes utat.
4. A paraméter nélküli APPEND a képernyőre írja az éppen érvényes bejárási utakat.
5. Hálózaton installált egységen az APPEND nem használható.
6. Az APPEND segítségével használt utakon csak olvasni lehet!
7. Szemben a PATH-tal, amely a bejárási utak mentén levő file-ok közül csak a .COM, .EXE és .BAT kiterjesztésűeket találja meg, az APPEND-del tetszőleges kiterjesztésű file elérhető.

5. DOS: a fix lemez kezelése

A PC DOS, néhány korai verziójától eltekintve, támogatja a fix lemez kezelését. Ha a fix lemezre írni vagy arról olvasni szeretnénk, akkor a meghajtó azonosítóját kell megadni. Ez általában az

- első fix lemez esetén a C,
- a második fix lemez esetén D.

A fix lemez négy különböző méretű területre osztható fel. A területeket partíciónak nevezzük. A partíciók mindegyikét egy-egy operációs rendszer használhatja. Lehetőségünk van viszont arra is, hogy az egész területet egy operációs rendszer kezelje. A különböző operációs rendszerek mindegyike tartalmaz olyan komponenst, amely a partíciók kezelését végzi. A PC DOS rendszerben ez az FDISK program.

A DOS 3.20-as és korábbi változatai nem tették lehetővé, hogy a DOS több partíciót is használjon. Erre nem is volt eleinte szükség, hiszen a lemezegységek mérete kicsi volt. A lemezegységek kapacitásának növelése azonban megkívánta, hogy több partíciót is használhasson a DOS . A partíciók szerkezete ugyanis olyan, hogy maximum 32 Mbyte információ fér rá. Így az ennél nagyobb egységek 'felesleges' kapacitását a DOS maga már nem tudta kezelni.

A DOS 3.30-as verziója kétféle DOS partíciót ismer. Az első az ún. elsődleges DOS partíció, a másik az ún. kiegészítő DOS partíció. Angol elnevezésük: primary, ill. extended. A fixlemezen egy elsődleges és egy kiegészítő DOS partíció rendelhető. A kiegészítő DOS partíció területét ezután logikai egységekre oszthatjuk. Egy logikai egység maximum 32 Mbyte lehet. Így tetszőleges méretű fix egységet képes a DOS kezelni, azzal a megkötéssel, hogy a kiegészítő DOS partícióban levő logikai egységek maximum 32 Mbyte naagyságúak lehetnek.

5.1 FDISK - fix lemez kialakítása

Az FDISK program lehetővé teszi, hogy a fix lemezen

- DOS-partíciót előállítsunk,
- DOS-partíciót aktivizáljunk,
- DOS-partíciót töröljünk,
- a DOS-partíció jellemzőit kiírassuk,
- áttérjünk a másik fix lemezre.

Verzió: 2.11 - 3.30
A program azonosítója: FDISK.COM

Indítása tehát úgy történik, hogy az FDISK kulcsszót beírjuk. Az indításkor a programnak nem kell paramétereket megadni. A program a képernyőre kiírja a választható lehetőségeket (a menüt). A megfelelő lehetőség kiválasztása a mellé írt szám megadásával történik.

Az FDISK programhoz az ESC billentyű lenyomásával térhetünk vissza, ismételt lenyomásával pedig a DOS rendszerhez.

A fix lemez szerkezete

A fix lemez cilinderekre van felosztva. A cilinderek száma és mérete a fix lemez típusától függ. A 10 Mb-os lemez 305 cilindert tartalmaz.(A 20 Mbyte-os általában 611 cilindert.) Egy cilinder kapacitása 32768 byte.

Az első cilinder sorszáma: 000, az utolsóé pedig: 304.

Az aktív partíció azt a DOS-partíciót jelöli, amelyről a DOS rendszer betölthető. Abban az esetben, ha két fix lemez van a rendszerben, csak az elsőn lehet aktív partíciót kijelölni. A DOS rendszert csak az első fix lemezről lehet betölteni.

A következőkben ismertetjük a különböző választási lehetőségek lépéseit. A menü pontjai és azok jelentése a következő:

1. Create DOS Partition	DOS partíció létrehozása
2. Change Active Partition	az aktív partíció kijelölése
3. Delete DOS Partition	DOS partíció törlése
4. Display Partition Data	a partícióadatok kiíratása
5. Select Next Fixed Disk Drive	a következő fix lemezegység kiválasztása

5.1.1. DOS-partíciók előállítása

Elsődleges DOS-partíció:

1. lépés: Az FDISK indítása. Ezután a képernyőn megjelenik a főmenü.

2. lépés: Az 1 választásával és beírásával, majd az < ENTER > megnyomásával a DOS-partíció előállítását választjuk ki.

3. lépés: A program megkérdezi, hogy elsődleges vagy kiegészítő DOS partíciót kívánunk-e létrehozni. Most az 1-es pontot kell választanunk. A DOS 3.20-as vagy korábbi verzióiban ez a kérdés kimarad, azokugyanis nem képesek csak az elsődleges DOS partíció kialakítására.

Ekkor a DOS újabb üzenetet küld, amelyben arról kér információt, hogy a teljes fix lemezt a DOS számára tartjuk-e fenn, vagy több partícióra (legfeljebb 4) szeretnénk osztani. (Megkérdezi: "Do you wish to use the entire fixed disk for DOS (Y/N)" - kívánjuk-e a teljes lemezt fenntartani a DOS számára - N-t válaszolunk, ha több partíciót akarunk.) Ha az egész fix lemezt a DOS alatt akarjuk használni, akkor csak az üzenet végét jelző billentyűt (< ENTER >) kell lenyomni.

4. lépés: A DOS újraindítása az A: vagy a B: meghajtóról egy tetszőleges billentyű lenyomásával történik. Ha a DOS-partíciót a második fix lemezen hoztuk létre, akkor a rendszert nem kell újraindítani.

5. lépés: A fix lemez DOS partíciójának előkészítése a file-ok befogadására a

```
FORMAT C:/S/V
```

vagy

```
FORMAT C:/V
```

paranccsal történik.

A C: a fix lemez azonosítója. A /S paraméter hatására a rendszer-file-okat is átmásolja, a /V paraméter megadása esetén pedig legfeljebb 11 jelből álló kötetazonosítót helyez el a lemezen a rendszer.

Ha az egész DOS rendszert el akarjuk helyezni a lemezen, akkor a parancs a következő:

```
COPY *.* C:
```

Az aktuális meghajtóról valamennyi file átkerül a fix lemezre.

6. lépés: A DOS rendszer indítása fix lemezről. Az A: meghajtóból kivesszük az eredeti DOS lemezt. A <Ctrl-Alt-Del> együttes lenyomásával már a fix lemezről indíthatjuk újra a rendszert.

Megjegyzés:

Ebben az esetben az A: meghajtóban nem lehet DOS rendszert tartalmazó lemez. A számítógép először az A: meghajtóról próbál rendszert tölteni. Ha ez eredménytelen, csak ezután fordul az első fix lemezhez.

A dátum és az idő megadása után már új egységazonosító jelenik meg (C>), amely az első fix lemezt azonosítja. A DOS aktuális meghajtónak most a C: meghajtót tekinti.

Ha már a fix lemezt más rendszerek alatt használtuk, akkor informatív üzenet jelenik meg a partíciókról. Ez tartalmazza a partíció

- azonosítóját (1.-4.),
- állapotát (A-aktív, N-nem aktív),
- típusát (DOS partíció-e),
- kezdetét (kezdő cilinder),
- végét (utolsó cilinder),

valamint a teljes lemezterület méretét cilinderben, a még rendelkezésre álló szabad terület méretét és kezdő címét. Abban az esetben, ha két fix lemez van a rendszerben, a meghajtó azonosítóját is kiírja.

Ha a lemezen még nincs 4 partíció kijelölve és szabad terület is van, akkor ezen a területen létrehozhatunk egy DOS-partíciót.

Kiegészítő DOS partíció (csak 3.30-as verziótól)

Ha a lemezen nincs még négy partíció kijelölve, és a meglevő partíciók egyike sem kiegészítő partíció, akkor lehetőségünk van annak megadásaira. A művelet egyes lépései a következők:

1. lépés: Az FDISK indítása. Ezután a képernyőn megjelenik a főmenü.

2. lépés: Az 1 beírásával és az < ENTER > megnyomásával a DOS partíció előállítását választjuk.

3. lépés: A program megkérdezi, hogy elsődleges vagy kiegészítő DOS partíciót kívánunk-e létrehozni.Most a 2-es pontot választjuk, a kiegészítő DOS partíció megadását.

Ekkor a DOS újabb üzenetet küld, melyben jelzi, hogy még hány cilinder áll rendelkezésre. A program válaszul a maximálisan rendelkezésre álló cilinderek számát kínálja. Egyébként tetszőleges - ennél kisebb - értéket adhatunk meg.

Újabb üzenet közli, hogy a kiegészítő DOS partíciót létrehozta a rendszer.

4. lépés: Logikai egységek hozzárendelése a kiegészítő DOS partícióhoz. Az < ESC > megnyomása után újabb üzenetet kapunk, hogy még nincs logikai egység definiálva. A program az előző logikai egység nagyságát kéri, s válaszul a teljes kiegészítő DOS partíciót kínálja. Tetszőleges számú és nagyságú logikai egységet létrehozhatunk, ezek összmérete meg kell, hogy egyezzen a kiegészítő DOS partíció méretével.

5.1.2. Aktív partíció változtatása

Ezt a lehetőséget akkor választjuk, ha az operációs rendszert egy másik partícióból szeretnénk indítani.

1. lépés: Az FDISK indítása. Ezután a képernyőn megjelenik a választható lehetőségek listája.

2. lépés: A 2 megadásával az aktív partíció változtatását választjuk ki. Ezután a DOS informatív üzenetet küld a partíciókról, és bekéri az aktivizálandó partíció azonosítóját. Az < ESC > billentyű lenyomásával az FDISK főmenühöz térhetünk vissza. Az < ESC > ismételt lenyomásával pedig a DOS-hoz.

A számítógép továbbra is az eredeti partícióval dolgozik. Ha az új aktív partícióról akarjuk a rendszert indítani,akkor a < Ctrl-Alt-Del > billentyűkombinációval újra kell tölteni a fix lemezről a rendszert. Ilyenkor az A: meghajtóban természetesen nem lehet lemez.

Megjegyzés: Aktív partíció csak az első fix lemezen lehet.

5.1.3. DOS-partíció törlése

Ez a file-ok törlését is jelenti. Ezért, ha ezt a lehetőséget akarjuk választani, előtte a file-okat ajánlatos kimenteni. Az aktív DOS-partíciót nem lehet törölni, előtte át kell állítani (2. választási lehetőség).

A végrehajtás lépései hasonlítanak az aktív partíció változtatásánál megismertekhez, de értelemszerűen a 2. lépésben a 3 megadásával választható ki a DOS-partíció törlése funkció.

5.1.4. A partícióadatok kiíratása

Ezt a lehetőséget akkor választjuk, ha a fix lemez felosztásáról szeretnénk információt kapni.

A képernyőn a következő információk jelennek meg:

- a partíció azonosítója,
- állapota (aktív - nem aktív),
- típusa,
- kezdete,
- vége,
- mérete.

Megkapjuk a teljes lemezterület méretét (cilinderben). A fix lemez meghajtója csak abban az esetben kerül kijelzésre, ha a rendszerben több fix lemez van.

Ha a DOS 3.30-as vagy későbbi verzióit használjuk, akkor a program megkerdezi, hogy a kiterjesztett DOS partícióhoz rendelt logikai egységek adatait is kijelezze-e. Ha < 4 > -et, majd < ENTER > nyomunk le, akkor a képernyőre kerülnek ezek az adatok is.

Az üzenet utolsó sorában az FDISK főmenühöz való visszatérés módja van leírva: < ESC > billentyű lenyomása.

A kiírás nem tartalmaz információt a fix lemez szabad területéről.

5.1.5. A következő fix lemez kiválasztása

Ezt a lehetőséget csak akkor választhatjuk, ha a rendszerben legalább két fix lemez van. Ha ezt választottuk, akkor az FDISK "átlép" a másik fix lemezre. A képernyőn megjelenő FDISK főmenüben a pillanatnyi fix lemez meghajtó azonosítója megváltozik. Például, ha kezdetben 1 volt, az "átlépés" után 2 lesz. Ez lehetővé teszi, hogy az FDISK ismételt behívása nélkül két fix lemez felosztását is elvégezzük. A DOS rendszerhez az < ESC > billentyű lenyomásával térhetünk vissza.

5.2 Fix lemez archiválása

A fix lemez teljes, vagy meghatározott file-jainak kimentését floppy-ra, és arról visszatöltését a BACKUP és a RESTORE parancs segítségével lehet elvégezni. A file-ok speciális formában kerülnek a floppy-ra, így ezek nem használhatók fel, csak a fix lemezre való visszatöltés után. Ha a fix lemez tartalma nem fér el egy floppy lemezen, a DOS folytatás kötetet kér. Természetesen ezeket a visszatöltés során is tudja kezelni.

5.2.1 BACKUP - file-ok archíválása

A fix vagy floppy lemez teljes tartalmát, vagy meghatározott file-jait menti floppy-ra vagy fix lemezre.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

BACKUP *d:*[*út*][*file-név*[*.kit*]] *d:*[/S][/M][/A][/D:*hh-nn-éé*]
[/T:*óó:pp:mm*][/F][/L[:[*d:*][*út*]*file-név*[*.kit*]]]

A paraméterek jelentése:

Az első paramétersor a mentésre kijelölt file-okat írja le. A második paramétersor azt a meghajtót azonosítja, amelyre a mentés történik. A floppy lemezeket előzőleg meg kell formázni.

A forrás- és a cél lemezegységnek (d:) különbözőknek kell lennie. Mind floppy egység, mint Winchester tetszőleges kombinációban szerepelhet forrásként és célként.

Azon lemezegységet, amelyik nem az aktuális, meg kell adni! (d:)

/S	megadásával a megadott, illetve ha nem adtuk meg, az aktuális tartalomjegyzékhez tartozó összes file kimentésre kerül (az összes, ezen tartalomjegyzékben levő altartalomjegyzékbeli file-ok is)
/M	megadása esetén csak azok a file-ok kerülnek mentésre, amelyeket az utolsó archíválás óta módosítottunk
/A	megadása esetén a cél-lemezen már meglévő file-ok nem törlődnek, egyébként csak a kimentett file-ok lesznek a cél-lemezen.
/D:*hh-nn-éé*	megadásával egy időpontot jelölünk ki, amelytől kezdve felvitt, illetve módosított file-ok kerülnek lementésre.
/T:*óó:pp:mm*	megadásakor csak azok a file-ok kerülnek mentésre, amelyek az adott naptól és időponttól kezdve valamikor módosításra kerültek, vagy akkor hozták létre.
/L	segítségével a lemezre LOG file-t írhatunk. Ennek a neve BACKUP.LOG lesz. Az első rekord a mentés dátumát és időpontját tartalmazza. A következő rekordok tartalma: a mentéslemez sorszáma, és minden file teljes neve.
/L:[*d:*][*út*]*file név*[*.kit*]]	segítségével módosíthatjuka LOG file nevét.

Megjegyzés:

1. A file létrehozásakor és minden módosításakor a tartalomjegyzékbe bekerül a napi dátum és az idő is. Ezért lényeges, hogy a rendszer indításakor megadjuk a pontos időt és a dátumot.
2. A PROPOS-16 operációs rendszerben /C paraméterrel másolás utáni ellenőrzés kérhető.
3. A BACKUP és a RESTORE parancsok az IBM PC DOS, illetve az MS DOS esetén nem ugyanazok. Az egyes verziók sem kompatibilisek egymással. A fenti formátum az IBM PC DOS BACKUP parancsa, az MS DOS ettől eltér:

BACKUP [*d*:][*út*][*file-név*[.*kit*]][*d*:][/S][/M][/A][/D:*hh-nn-éé*] [/P][/T:*óó:pp:mm*][/L[:*file-név*]]

Az új paraméter használata a következő:

/P a megadjuk ezt a paramétert, akkor annyi file-t másol egyetlen hajlékonylemezre, amennyit csak tud. Ebhez esetenként egy altartalomjegyzéket is létrehoz. (Ennek az opciónak a használata nem kompatibilis az IBM PC DOS BACKUP parancsával!)

4. Az /L ésa /T opciók a DOS 3.20-as és korábbi verziókban nem használható! Az MS DOS rendszerben azonban ezek a paraméterek is használhatók.

A BACKUP működése:

Ha nem adjuk meg a /A paramétert, akkor a floppy vagy egyéb cél lemez valamennyi file-ját törli, mielőtt a forrás lemez file-jainak a mentését elkezdené. Ha floppy-ra mentünk, és a lemez megtelt, akkor felkéri a felhasználót, hogy cserélje ki a floppyt. Az archíválásra kerülő file-ok azonosítóját kijelzi.

A BACKUP a következő módon állítja a befejezési kódot:

0 - Normális befejezés.
1 - A file nem található.
2 - File megosztási probléma miatt néhány file nem került elmentésre.
3 - Felhasználó kényszerítette a mentés befejezését (<Ctrl-Break>).
4 - Permanens I/O hiba.

Ezt a kódot a DOS parancsokból képezhető úgynevezett parancs file-okban felhasználhatjuk a parancsok feltételes végrehajtására (lásd 7. fejezet **IF ERRORLEVEL** parancs utasításforma).

Megjegyzés:

1. A file-ok kijelölésénél a globális név-jelek használhatók.
2. Ha az utat nem definiáljuk, akkor az aktuális tartalomjegyzék file-jai kerülnek mentésre.
3. A BACKUP paranccsal kimentett file-ok normál körülmények között nem használhatók. Felhasználás előtt a RESTORE paranccsal kell visszatölteni ezeket.
4. A parancs minden mentési lemezen létrehoz egy BACKUPID.@@@ file-t, amelyik a lemez sorszámát tartalmazza.

Példa:

1. Ha a következőt adjuk be:

```
BACKUP C:*.TXT A:
```

akkor a C: meghajtó aktuális tartalomjegyzékének valamennyi .TXT kiterjesztésű file-ja az A: meghajtóban levő floppy(k)ra kerül.

2. A következő paranccsal a C: meghajtó valamennyi file-ja mentésre kerül az A: meghajtón lévő lemezre.

```
BACKUP C:\ A:/S
```

A \ (backslash) a főtartalomjegyzéket jelöli ki, a /S paraméter pedig az összes altartalomjegyzéket és azok file-jait.

5.2.2 RESTORE - file-ok visszatöltése

Az előzőleg BACKUP paranccsal mentett file-ok visszatöltését végzi. Lehetőség van valamennyi, vagy meghatározott file-ok visszatöltésére.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

RESTORE *d:* [*d:*][*út*][*file-név*[.*kit*]][/S][/P][/B:*dátum*][/A:*dátum*][/E:*hh-nn-éé*]
[/L:*óó-pp-mm*] [/M][/N]

A paraméterek jelentése:

Az első paraméter azt a meghajtót jelöli, amelyről a visszatöltés történik. A második paraméter a cél lemez file-jait azonosítja.

/S	paraméter megadásával a mentett file-ok valamennyi altartalomjegyzékbe visszatöltésre kerülnek.
/P	paraméter megadása esetén a RESTORE üzenetet küld, mielőtt visszatöltené azokat a file-okat, amelyek az utolsó mentés óta nem változtak vagy csak olvasásra hozzáférhetők (Read Only). A két rendszer-file: IBMBIO.COM és az IBMDOS.COM is védett file.
/B	dátum megadásával csak a szóbanforgó dátumkor vagy előtte módosított file-okat tölti vissza
/A	dátum megadásakor csak a szóbanforgó dátum idején vagy az után módosított file-okat tölti vissza
/E	idő megadásakor csak az adott időpontban vagy azt megelőzően módosított file-okat tölti vissza
/L	idő megadásakor csak az adott időpillanatban vagy azt követően módosított file-okat tölti vissza
/M	esetén csak azokat a file-okat tölti vissza, amelyeket az utolsó elmentés óta módosítottunk
/N	esetén csak azokat a file-okat tölti vissza, amelyek nem léteznek a cél lemezen.

Megjegyzés:

1. Csak azok a file-ok tölthetők vissza, amelyeket előtte BACKUP paranccsal mentettünk le.
2. Ha az út paramétert nem adjuk meg, akkor a file-ok az aktuális tartalomjegyzéknek megfelelően kerülnek visszatöltésre.
3. Ha file-azonosítókat nem adunk meg, akkor valamennyi mentett file visszatöltésre kerül.
4. A globális név-jelek használhatók.
5. Ha a mentés több floppy-ra történt, a RESTORE ellenőrzi, hogy a felhasználó a lemezeket megfelelő sorrendben helyezte-e el. Ha nem a megfelelő folytatás kötetet tette fel, akkor üzenetben felkéri a cserére.
6. Vigyázat! Ha a rendszer-file-okat (IBMBIO.COM és IBMDOS.COM) is lementettük BACKUP-pal, ezeket NE töltsük vissza a fix lemezre, mert az nem lesz boot-olható! Ilyenkor a /P paramétert használjuk és a két rendszer-file-nál N-et válaszoljunk az engedélyezést kérő üzenetre.
7. A RESTORE a következő módon állítja a befejezési kódot:

   0 - Normális befejezés.
   1 - A file nem található.
   2 - File megosztási probléma miatt néhány file nem került visszatöltésre.
   3 - Felhasználó kényszerítette a mentés befejezését (<Ctrl-Break> ill. <ESC>).
   4 - Permanens I/O hiba a visszatöltés alatt.

   Ezt a kódot a DOS parancsokból képezhető úgynevezett parancs file-okban felhasználhatjuk a parancsok feltételes végrehajtására (lásd 7. fejezet IF ERRORLEVEL utasításforma).

Példa:

1. A következő parancs megadásával minden .TXT kiterjesztésű mentett file visszatöltésre kerül az A: meghajtóról az első fix lemezre:

```
RESTORE A: C:*.TXT
```

2. Ha valamennyi mentett file-t szeretnénk visszatölteni, akkor a következőt kell megadni:

```
RESTORE A: C:\/S/P,
```

illetve

```
RESTORE A: C: /S /P
```

(ha a rendszerfile-okat is lementettük.)

3. Ha egy megadott altartalomjegyzék valamennyi mentett file-ját szeretnénk visszatölteni, akkor a következőt adjuk meg:

```
RESTORE A: C:\ALTART1\*.*
```

6. A rendszer működését befolyásoló parancsok és kapcsolók

Ebben a fejezetben azokat a parancsokat ismertetjük, amelyekkel a rendszer állapotát változtathatjuk meg, illetve amelyek a rendszer állapotáról adnak információt. Így például:

- a képernyő törlése,
- a dátum és az idő megadása,
- a DOS verziójának lekérdezése,
- a < Break > billentyű figyelése,
- periféria-hozzárendelések és működésmódjuk megváltoztatása,
- a klaviatúra kiválasztása,
- grafikus jelek kezelése.

6.1 COMMAND - Újabb parancsfeldolgozó program elindítása

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

COMMAND [*d*:][*út*][/P][/C *karaktersorozat*][/E:*xxxx*]

A paraméterek jelentése:

A DOS az adott meghajtón és '*út*'-on keresi a parancsfeldolgozót, és betölti a memóriába. Ha a /P paramétert megadjuk, a felhasználói parancsprocesszor a DOS újraindításáig a memóriában marad. A /C paraméterrel a parancsprocesszornak a megadott *karaktersorozat*ot adhatjuk át, majd az automatikusan visszatér a rendszer eredeti parancsfeldolgozójához. Ha a /P és a /C paramétereket egyszerre adjuk meg, a /P-t a rendszer figyelmen kívül hagyja. /E:*xxxx* egy decimális szám, amelyik megadja a programkörnyezet (environment) nagyságát. Ez az érték a 160-32768 intervallumba kell hogy essen. Ez a paraméter csak a DOS 3.20 és későbbi verzióiban használható. (Az environmentről lásd 15.3 paragrafust.)

Például:

COMMAND /C DIR C:

Betöltődik a másodlagos parancsprocesszor és a DIR C: parancsot végre is hajtja, majd törlődik a memóriából.

Megjegyzés:

1. A felhasználói parancsprocesszor betöltése és használata után, ha nem adtuk meg sem a /P, sem a /C paramétert, a meghívó programhoz az EXIT paranccsal lehet visszatérni.

2. Ismét felhívjuk a figyelmet, hogy a DOS újraindítása után ismét a standard COMMAND.COM lesz a parancsprocesszor.

3. A DOS ezen tulajdonságát használja fel sok program DOS parancsok kiadására és programok indítására.

   A felhasználói programok is hívhatják az új parancsprocesszort. A felhasználói program paramétereket is átadhat az új parancsprocesszornak, amit az úgy hajt végre, mintha ezt a karaktersorozatot az aktuális input egységről olvasta volna.

A felhasználói parancsprocesszor hívásának lépései:

1. A memóriában 17 Kbyte szabad területet kell biztosítani az új parancsprocesszornak és paramétereinek. Meg kell győződni, hogy van-e még ekkora szabad terület.

2. Az új parancsprocesszornak fel kell építeni a paraméter mezőt. Ennek formája a következő:

1. byte tartalma:	a karaktersorozat hossza,
xx byte tartalma:	a karaktersorozat,
utolsó byte tartalma:	hexadecimális 0D (decimális 13), ami a "kocsivissza" (Carriage Return) kódja.

   Például a következő, assembly nyelvű utasítással építhető fel egy paraméter mező:

```
DB 17,"\C:DISKCOMP A: B:",13
```

   Ez azt eredményezi, hogy a felhasználói parancsprocesszor betöltése után az végrehajtja a DISKCOMP programot.

3. Az EXEC függvényhívással (hexadecimális 4B) az új parancsprocesszor kerül végrehajtásra. Ez a parancsprocesszor-váltás tükröződik a programkörnyezetben is, amelynek címe PSP + 26, ez a cím egyben a COMSPEC mező kezdetére mutat. (A programkörnyezet, a PSP - program szegmens prefix - és a COMSPEC fogalmát a 15. fejezetben, a program szegmens leírásánál ismertejtük.)

4. A másodlagos parancsprocesszor "örökli" az első PROMPT parancsban megadott készenléti jelet. Visszatéréskor ez nem történik meg.

6.2 EXIT - visszatérés az elsődleges parancsfeldolgozóhoz

A COMMAND paranccsal meghívott másodlagos parancsfeldolgozóból való visszatérésre biztosít lehetőséget az EXIT parancs.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

```
EXIT
```

Megjegyzés:

1. A parancsnak csak akkor van hatása, ha a COMMAND parancs kiadásakor sem a /S, sem a /P paramétert nem adtuk meg.

6.3 CLS - képernyő törlése

Törli a képernyő tartalmát. A kurzor a bal-felső pozícióba (home) kerül. A háttér és a keret színe nem változik meg.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

CLS

Megjegyzés:

Ha a képernyő grafikus módra van kapcsolva, csíkozott lesz. Ilyenkor a képernyőt normál módba a MODE paranccsal lehet visszakapcsolni.

6.4 DATE - dátum megadása

A parancs lehetővé teszi a napi dátum megadását és változtatását. A pontos dátum megadása azért lényeges, mert a file-ok létrehozásakor, módosításakor az is bekerül a tartalomjegyzékbe a file azonosítójával együtt.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

DATE [*hh-nn-éé*] | [*nn-hh-éé*] | [*éé-hh-nn*]

A paraméterek jelentése:

hh a hónap sorszáma, értéke: 1-12.

nn a hónapon belüli nap sorszáma, értéke: 1-31.

éé az év azonosítója, amely két formában adható meg:

- ha két számjegyet adunk meg, akkor 19-t feltételez, értéke ekkor: 80-99,
- négy számjegyet is megadhatunk, ekkor az értéke: 1980-2099.

Megjegyzés:

1. A paramétereket elválasztó jel / (slash) is lehet.
2. Ha érvénytelen paramétert adunk meg, akkor a rendszer ismételten kéri az új karaktersorozatot.

3. Ha csak a DATE kulcsszót írjuk be, akkor a DOS kiírja a rendszer-dátumot, majd kéri az aktuális dátumot. Ha nem akarjuk változtatni, akkor csak az üzenet végét jelző billentyűt (< ENTER >) kell lenyomni.

4. A dátum megadási módja a COUNTRY konfiguráció, illetve a SELECT DOS parancs használatától függ.

Példa:

Adjuk meg az új dátumot, ha az 86.03.07.

```
C>DATE

Date is: Thu 6-3-1986
Enter new date: 7/3/86
```

6.5 TIME - idő megadása

A parancs lehetővé teszi a pontos idő megadását, változtatását. A rendszer-idő a rendszer-dátumhoz hasonlóan bekerül a file létrehozásakor, módosításakor a tartalomjegyzékbe.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

TIME [*óó*[:*pp*[:*mm*[.*xx*]]]]

A paraméterek jelentése:

óó	az óra megadása, értéke: 0-23.
pp	a perc megadása, értéke: 0-59.
mm	a másodperc megadása, értéke: 0-59.
xx	a századmásodperc megadása, értéke: 0-99.

Megjegyzés:

1. Ha érvénytelen paramétereket adunk be, akkor a rendszer ismételten kéri az új időt.

2. Ha a századmásodperc részt nem adjuk meg, akkor 00-t feltételez.

3. Ha csak a TIME kulcsszót adjuk meg, akkor a DOS kiírja a rendszeridőt, majd kéri a pontos időt. Ha nem akarjuk változtatni, akkor csak az üzenet végét jelző billentyűt (< ENTER >) kell lenyomni.

Példa:

Adjuk meg a pontos időt, ha az: 8:10:37.00

```
C>TIME

Time is: 00:03:54.65
Enter new time: 8:10:37
```

6.6 VER - DOS-verzió lekérdezése

A parancs kiírja a DOS verziószámát.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

```
VER
```

Megjegyzés:

A DOS-verzió a következő információkat tartalmazza:

- a forgalmazó megnevezése,
- a főverziószám,
- a főverzión belüli változat.

Ha a parancsot az IBM DOS rendszere alatt adjuk ki, akkor IBM Personal Computer DOS Version 3.30 jelenik meg a képernyőn, az SZKI PROPOS-16 rendszerében pedig SZKI PROPOS-16 Ver. 3.xx

6.7 ASSIGN - periféria-hozzárendelések megváltoztatása

A parancs lehetővé teszi, hogy egy másik meghajtót jelöljünk ki a lemezes I/O műveletekhez.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

```
ASSIGN [x[ = ]y [...]]
```

A paraméterek jelentése:

Az *y* meghajtót *x*-szé nevezhetjük át. A meghajtó megadása után nem kell kettőspontot (:) tenni.

Megjegyzés:

1. Ha a parancsot paraméterek nélkül adjuk meg, akkor az eredeti meghajtó hozzárendelések visszaállnak.

2. A parancs alkalmazása csak olyan felhasználói programok futtatása előtt ajánlott, amelyekbe "be van építve", hogy egy megadott meghajtót használjanak, de az most nem áll rendelkezésünkre.
3. Ez a parancs nem az input vagy output átirányítását jelenti, hanem a meghajtó ideiglenes átnevezését.
4. A FORMAT, DISKCOPY és DISKCOMP parancsok figyelmen kívül hagyják a parancs hatását.
5. A parancsot nem lehet együtt használni a BACKUP, RESTORE, LABEL, JOIN, SUBST és PRINT parancsokkal.

Példa:

1. A következő példában az A: és a B: meghajtóról valamennyi I/O műveletet a fix lemezre irányítunk.

```
ASSIGN A = C B = C
```

2. A következő parancs, az

```
ASSIGN
```

az eredeti hozzárendeléseket visszaállítja.

6.8 BREAK - < Break > billentyű figyelése

Ez a parancs lehetővé teszi, hogy a rendszer a DOS-funkciók elvégzése alatt figyelje, hogy a < Ctrl-Break > billentyűt lenyomtuk-e.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

```
BREAK [ON | OFF]
```

A paraméterek jelentése:

Ha az ON paramétert adjuk meg, akkor a < Ctrl-Break > bármilyen DOS funkció hívásakor érvényre jut.

Ha az OFF paramétert adjuk meg, akkor a < Ctrl-Break > csak akkor kerül érvényre, amikor a végrehajtás alatt álló program a billentyűzetre, a képernyőre, a nyomtatóra vagy valamelyik vonali adapterre kér I/O műveletet. Ez a BREAK kapcsoló alapértelmezése.

Ha nem adunk meg paramétert, akkor aktuális értéke kerül kijelzésre.

Megjegyzés:

1. A BREAK értéke mindaddig érvényben marad, amíg egy másik BREAK paranccsal azt át nem állítjuk.

2. A BREAK ON megadása csak akkor indokolt, ha egy olyan programot futtatunk, amely főleg file I/O műveleteket végez, de a felhasználóval alig kommunikál (nincs/kevés a billentyűzet input, ill. képernyőre írás).

6.9 MODE - perifériák működésmódjának beállítása

A parancs lehetőséget ad a képernyő, a nyomtató és az aszinkron kommunikációs adapterek működésmódjának a beállítására. Alkalmazásának területei:

- a képernyő üzemmódjának beállítása,
- a nyomtatási formátum meghatározása,
- az aszinkron kommunikációs adapter üzemmódjának a beállítása,
- a párhuzamos illesztésű nyomtató outputjának átirányítása az aszinkron kommunikációs adapterre.
- a kódlapozás paramétereinek beállítása,

Ha a MODE parancsot nem a képernyő üzemmódjának beállítására használjuk, akkor a DOS mérete megnő.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

1. **MODE LPT#[:][*n*][,[*m*][,P]]**
2. **MODE *n***
3. **MODE [*n*],*m*[,T]**
4. **MODE COM*n*[:]*baud*[,[*paritás*][,[*adatbitek*][,[*stopbitek*][,P]]]]**
5. **MODE LPT#[:]=COM*n***
6. **MODE *egység* CODEPAGE PREPARE=((*kp*)[*d:*][*út*]*file-név*[.*kit*])**
7. **MODE *egység* CODEPAGE PREPARE=((*kp-lista*)[*d:*][*út*]*file-név*[.*kit*])**
8. **MODE *egység* CODEPAGE SELECT=*kp***
9. **MODE *egység* CODEPAGE [/STATUS]**
10. **MODE *egység* CODEPAGE REFRESH**

Az egyes kulcsszavakat a következőképpen rövidíthetjük:

• CP	CODEPAGE helyett
• PREP	PREPARE helyett
• SEL	SELECT helyett
• REF	REFRESH helyett
• STA	STATUS helyett

I. Alkalmazás - A nyomtató üzemmódjának beállítása.

A parancs formája:

MODE LPT#[:][*n*][,[*m*][,P]]

A paraméterek jelentése:

#	Értéke 1, 2, illetve 3 lehet.
n	Az értéke 80, vagy 132 lehet,az egy sorba nyomtatható karakterek maximális száma.
m	Az értéke 6 vagy 8 lehet, a sorok száma 1 coll (= 2.54 cm) távolságon.
P	A folyamatos ismétlést állítja be idő-túllépés (time-out) hiba esetén. Ha P-t adtunk meg, és már nem akarjuk, hogy a rendszer a time-out hibánál újra próbálkozzon, a P hatását úgy lehet megszüntetni, hogy a nyomtatót újra inicializáljuk a P megadása nélkül.

Megjegyzés:

A nyomtatóra bekapcsoláskor a következő alapértelmezések élnek: l = 80; m = 6; tehát 80 karakter soronként és 6 sor/coll.

Példa:

A következő paranccsal a nyomtató üzemmódjának valamennyi paraméterét állítjuk:

```
MODE LPT1:132,8,P
```

Tehát 132 karakter/sor; 8 sor/coll; idő-túllépés (time-out) esetén pedig folyamatos ismétlés.

II. Alkalmazás - A színes/grafikus képernyő adapter üzemmódjának a beállítása, átváltása.

A parancs formája:

1. **MODE** *n*
2. **MODE** [*n*],*m*[,**T**]

A paraméterek jelentése:

n	Értéke: 40; 80; BW40; BW80; CO40; CO80; vagy MONO lehet.
40	A színes/grafikus adapter esetén az egy sorban kijelezhető maximális karakterek száma: 40.
80	A színes/grafikus adapter esetén az egy sorban kijelezhető maximális karakterek száma: 80.
BW40	Az aktív képernyő adaptert fekete-fehér módra kapcsolja át, a kijelzés soronként 40 karakter lesz.
BW80	Ez ugyanaz, mint a BW40, csak soronként 80 karakterrel.
CO40	Az aktív képernyő adaptert színes módra kapcsolja át.
CO80	Ez ugyanaz, mint CO40, csak soronként 80 karakterrel.
MONO	Az éppen aktív képernyő adaptert monokróm adapterre kapcsolja át. A képernyőre mindig 80 karakter írható ki soronként.
m	Az értéke **R** vagy **L** lehet. A képernyő kijelzésének jobbra vagy balra történő mozgatását jelenti. (R - jobbra, L - balra.)
T	A rendszer a képernyő beállításához teszt-képernyőformákat küld, amellyel a kijelzés ellenőrizhető. (Ez egyjegyű számokból álló sor.)

Megjegyzés:

1. A 40; 80; CO40; CO80 és a MONO értékek csak színes/grafikus adapter esetében használhatóak.
2. A képernyő kijelzése mindkét irányba eltolható:

   - 40 karakter/sor esetén 1 karakterrel,
   - 80 karakter/sor esetén 2 karakterrel.

   Ha a T paramétert megadjuk, akkor üzenetet kapunk, hogy megfelelő-e a képernyő. Ha az Y-t, adjuk meg, akkor a parancs befejeződik. Ha pedig az N-t, akkor a képernyő eltolását megismétli, és ismét üzenetet küld a rendszer.

Példa:

A következő parancs

```
MODE 80,R,T
```

esetén az üzemmód a következő:

- 80 karakter/sor,
- a képernyő kijelzését 2 karakterrel jobbra tolja,
- teszt-képernyőformát küld.

III. Alkalmazás - Az aszinkron kommunikációs adapter üzemmódjának beállítása.

A parancs formája:

MODE COM*n*[:]*baud*[,[*paritás*][,[*adatbitek*][,[*stopbitek*][,P]]]]

A paraméterek jelentése:

n	Értéke 1 vagy 2 (3.30 esetén még 3 vagy 4 is), az aszinkron kommunikációs adapter azonosítója.
baud	110; 150; 300; 600; 1200; 2400; 4800; 9600 (illetve 3.30 esetén 19200) átviteli sebesség. Az azonosítóhoz elég az első két karaktert megadni.
paritás	N - nincs paritás, O - páratlan, E - páros (alapértelmezés).
adatbitek	7 (alapértelmezés) vagy 8.
stopbitek	1 vagy 2. Ha az átviteli sebesség értéke 110 baud, akkor az alapértelmezés: 2, egyébként 1.

A P paramétert csak akkor kell megadni, ha az aszinkron adaptert egy soros illesztésű nyomtató esetében használjuk. Ha a P paramétert megadtuk, akkor az idő-túllépés (time-out) hiba esetére a folyamatos ismétlést állítottuk be. Ezt a <Ctrl>-<Break> billentyű lenyomásával szüntethetjük meg. A hibaüzeneteket úgy szüntethetjük meg, hogy a MODE parancsot kiadjuk a P paraméter nélkül.

Megjegyzés:

Az aszinkron adatátviteli-protokoll paramétereinek a megadása az adapter inicializálását is jelenti. Az átviteli sebesség értékét mindig meg kell adni.

Példa:

Ha a következő parancsot adjuk meg:

```
MODE COM1:12,N,8,1,P
```

akkor a következő értékekkel inicializáljuk az első aszinkron átviteli adaptert:

- átviteli sebesség: 1200 Baud,
- paritást nem ad,
- az adatbitek száma: 8,
- egy stopbit,
- idő-túllépés esetén ismétlés.

Az alapértelmezések felhasználásával ugyanez a parancs a következő formában is kiadható:

```
MODE COM1: 12,N,8,,P
```

IV. Alkalmazás - A párhuzamos illesztésű nyomtató átirányítása az aszinkron kommunikációs adapterre.

A parancs formája:

MODE LPT#[:] = COM*n*

A paraméterek jelentése:

#	1, 2 vagy 3, a nyomtató azonosítója,
n	1 vagy 2 (3.30 esetén még 3 vagy 4 is), az aszinkron adapter azonosítója.

Megjegyzés:

1. A párhuzamos illesztésű nyomtató outputjának átirányítása előtt az aszinkron kommunikációs adaptert inicializálni kell (III. Alkalmazás). Az inicializáló parancsnak a P paramétert is tartalmaznia kell.
2. Ha a MODE LPT#[:][n][,[m],P] parancsot kiadjuk (az n-dik nyomtató üzemmódjának beállítása), akkor az érvényteleníti az ugyanarra a nyomtatóra kiadott LPT# = COMn parancsot.

V. Alkalmazás - kódlapok előkészítése

A parancs formája:

MODE *egység* **CODEPAGE PREPARE =** ((*kp*)[*d:*][*út*]*file-név*[*.kit*])
MODE *egység* **CODEPAGE PREPARE =** ((*kp-lista*)[*d:*][*út*]*file-név*[*.kit*])

A paraméterek jelentése:

egység	a következőlogikai egységek valamelyike: CON, PRN, LPT1, LPT2 vagy LPT3.
kp	valamelyik kódlap sorszáma
kp-lista	kódlapok sorszámainak listája. Az egyes értékeket vesszővel kell elválasztani és az egész listát kerek zárójelek közé kell tenni. (A kódlap csak a 437, 850, 860, 863 vagy 865 értékek valamelyike lehet.)

[*d:*][*út*]*file-név*[*.kit*] a kódlapokat tartalmazó file neve. A rendszerhez az alábbi CPI kiterjesztésű file-okat kapjuk:

- 4201.CPI - IBM Proprinter
- 5202.CPI - IBM Quietwriter III. Printer
- EGA.CPI - EGA kompatibilis eszköz
- LCD.CPI - IBM LCD

Példák:

```
MODE LPT1:CODEPAGE PREP = ((850,865) 4201,CPI)
MODE LPT1:CP PREP = ((437)D:\4201.CPI)
MODECON:CP PREP = ((850,,437) EGA.CPI)
```

A harmadik példában a 850-es és 437-es kódlap közé egy üres lap került. Ennek az a hatása, hogy a CON: egységre előzőleg definiált második kódlap változatlan marad.

VI. Alkalmazás - Kódlap kiválasztása

Az V. alkalmazásban csak a memóriába töltjük azokat a kódlapokat, amiket használni akarunk. Ezek közül a használni kívántakat az aalábbi paranccsaal lehet kiválasztani:

MODE *egység* **CODEPAGE SELECT =** *kp*

A paraméterek jelentése:

egység	A következő logikai egységek valamelyike: CON, PRN, LPT1,LPT2 vagy LPT3.
kp	a használni kívánt kódlap sorszáma. A lehetséges értékek: 437, 850, 860, 863 vagy 865. Csak olyan értéket választhatunk, amelyet egy megelőző MODE PREP parancs listájában felsoroltunk.

Példák:

```
MODE CON: CP SEL = 850
MODE LPT1: CP SEL = 865
```

VII. alkalmazás - Kódlap sorszámának lekérdezése

MODE *egység* CODEPAGE [/STATUS]

A parancs az egységhez rendelt kódlapok számát és az éppen aktív kódlap számát a képernyőre írja. 'egység' a (CON,PRN,LPT1, LPT2 és LPT3 logikai egységek bármelyike lehet.

VIII. alkalmazás - kódlap újratöltése

MODE *egység* CODEPAGE REFRESH

A parancs újra olvassa az utolsó MODE CP PREP parancsban megadott file-t. Erre időnként szükség lehet, ha pl. valamiért kikapcsoltuk a nyomtatót. Az 'egység' paraméter a CON, PRN, LPT1, LPT2 es LPT3 logikai egység bármelyike lehet.

Példa:

```
MODE LPT1: CP PREP = ((860) D:\CPDIR5202.CPI)
...
MODE LPT1: CP REF
```

6.10 KEYB - nemzeti billentyűzet kiválasztása

A parancs lehetővé teszi, hogy a DOS BIOS billentyűzet-kezelő programját felülbíráljuk, és a tasztatúrának megfelelő programot töltsünk be.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája (3.20-as vagy korábbi verziónál):

KEYB*xx*

A parancsnak nincsenek paraméterei, de a kulcsszóban az *xx* helyére a következő karakterek kerülhetnek:

UK - angol,
FR - francia,
SP - spanyol,
GR - német,
IT - olasz.

A parancs formája (3.30 verziónál):

KEYB [*xx*[,[*yyy*],[[*d:*][*út*]*file-név*[.*kit*]]]]

A paraméterek jelentése:

xx	a használható billentyűzet kódok egyike.

yyy	a használandó kódlap. Ha nem adjuk meg, akkor a program az adott nemzeti billentyűzet kódhoz tartozó kódlappal dolgozik.
[*d:*][*út*]*file-név*[*.kit*]	a billentyűzet leíró file-t azonosítja. Ha nem adjuk meg, akkor a KEYB parancs az aktuális egység főtartalomjegyzékében levő KEYBOARD.SYS nevű file-t használja.

Megjegyzés:

1. A nemzeti billentyűzet kiválasztására szolgáló program 3.30-as verziója nem kompatibilis az előző verziókkal.
2. A DOS indítása után az aktuális klaviatúra az amerikai (feltéve hogy pl. az AUTOEXEC.BAT file-ban nem állítjuk át). Ha több KEYBxx parancsot adunk ki, akkor mindig újabb program töltődik a memóriába, de az előző sem törlődik.
3. A KEYBxx programból bármikor visszatérhetünk az amerikai billentyűzetre a <Ctrl-Alt-F1> lenyomásával.
4. Mindig az utoljára betöltött programot és az amerikai billentyűzet-kiosztást lehet csak elérni. A rendszer újraindításakor a betöltött tasztatúra-kezelő programok törlődnek.
5. Magyarországon több tasztatúra-kezelő program is megvalósítja a magyar ékezetes (szabvány írógép) billentyűkezelést. Közülük néhányat KEYBHU-nak neveznek.

6.11 GRAFTABL - grafikai jelek betöltése

A parancs segítségével a színes/grafikus adapter működéséhez egy karaktertáblázatot tölt be a memóriába. A táblázat a memóriában a PC DOS rendszer alatt 1392 byte-ot foglal el.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

GRAFTABL [437|860|863|865|/STATUS]

A paraméterek jelentése:

437	- USA (alapértelmezés)
860	- Portugál
863	- Kanadai (francia)
865	- Norvég vagy dán
/STATUS	hatására kiírja a kiválasztott kódlap számát.

Megjegyzés:

1. A parancs az 1F megszakítási vektort (interrupt vektor) változtatja meg, így a 128 és a 255 érték közötti ASCII-kódú jelek is kiírathatók a képernyőre, ha a színes/grafikus adaptert grafikus módba állítottuk be.

2. A parancs a kódtáblázat betöltése és a megszakítási vektor átállítása után a következő üzenetet adja:

   GRAPHIC CHARACTERS LOADED
   (A grafikus jelkészletet betöltötte.)

3. A parancs ismételt kiadása esetén a rendszer a következő üzenetet küldi:

   GRAPHIC CHARACTERS ALREADY LOADED
   (A grafikus jelkészletet már betöltötte.)

   Ez a parancs a Propos 3.0 és a Commodore PC-10, PC-20 operációs rendszerében nem áll rendelkezésre.

4. A DOS néhány verziójánál, ha a GRAFTABL parancsot a rendszer indítása után ismételten kiadjuk, nem kapunk üzenetet. A parancs ismételt kiadása után viszont a DOS mérete mindig mintegy 132 Kbyte-tal bővül (így a felhasználói szabad memória ennyivel csökken).

   A GRAFTABL a következő kódokkal tér vissza:

   0 - Nem volt előzően definiált kódlap, a megadott rezidensen a tárban maradt.
   1 - Volt már előzően definiált kódlap, ha az új attól eltérő, akkor az előző helyére kerül.
   2 - Nem volt előzőleg definiált kódlap és nem is keletkezett.
   3 - Nem megfelelő paraméter
   4 - Nem megfelelő DOS verzió.

6.12 GRAPHICS - grafikus nyomtatás lehetősége

Ez a parancs lehetővé teszi, hogy a képernyő grafikus tartalmát a nyomtatóra kiírassuk. A parancs kiadása után a DOS rezidens része a memóriában 688 byte-tal megnő. Ez a parancs csak akkor használható, ha a számítógép rendelkezik színes/grafikus adapterrel és monitorral.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

GRAPHICS [*nyomtató típusa*][**/R**][**/B**][**/LCD**]

A paraméterek jelentése:

/R	esetén ami a képernyőn fehér, az nem kerül nyomtatásra, ami fekete, az igen. Ha a paramétert nem adjuk meg, akkor a fehér fekete, a fekete pedig fehér lesz.
/B	megadásának csak színes nyomtató esetén van értelme, s hatására a háttér színét is nyomtatja a rendszer. Egyébként a háttér színe fehér lesz nyomtatáskor.

/LCD megadása után pontosan úgy nyomtat, ahogy az az IBM PC Convertible Liquid Crystal Display típusú képernyőn megjelenik.

A 'nyomtató típusa' lehetséges értékei a következőek:

COLOR1 - IBM színes nyomtató fekete szalaggal;
COLOR4 - IBM színes nyomtató RGB szalaggal;
COLOR8 - IMB színes nyomtató CMY szalaggal;
COMPACT - IBM Compact Printer;
GRAPHICS - IBM Personal Graphics Printer
THERMAL - IBM PC Convertible Printer és IBM Proprinter

Ha nem adjuk meg, az GRAPHICS-nek számít. Ezzel a fajtával kompatibilisek az EPSON FX sorozatú nyomtatók.

Megjegyzés:

A képernyő kinyomtatását a <Shift-PrtSc> billentyűk lenyomásával indíthatjuk.

Ha a képernyő grafikus módban van, a <Shift-PrtSc> minden egyes lenyomására a következő történik:

- 320 x 200-as színes/grafikus módban a képernyő tartalmát négy szürke árnyalattal a nyomtatóra írja,
- 640 x 200-as színes/grafikus módban a képernyő tartalmát 90 fokkal elforgatva (az óramutató járásával ellentétes irányban) a nyomtatóra írja.

A képernyő teljes kinyomtatása legfeljebb 3 percig tart (természetesen függ a nyomtató típusától is).

A képernyő tartalmának kinyomtatása programból a következő módon kezdeményezhető:

```
PUSH BP
INT 5
POP BP
```

7. Parancs file-ok

A DOS lehetővé teszi, hogy parancsból paraméterezhető "eljárásokat" írjunk, ezeket file-okba tegyük, és innen hajtassuk végre. E file-okat itt **parancs-file**-oknak fogjuk nevezni (az angol irodalomban: **batch-file**). **Minden parancs-file nevének kiterjesztése .BAT.** A parancs-file-okban levő "eljárások" "utasításai" DOS parancsok lehetnek, ezek végrehajtását az e fejezetben sorra kerülő úgynevezett DOS alparancsokkal vezérelhetjük. A parancs file-ok végrehajtása a file-név és az aktuális paraméterek megadásával kezdeményezhető.

A <Ctrl-Break> billentyű lenyomásával a végrehajtás alatt álló DOS parancs feldolgozása befejeződik. A rendszer üzenetet küld, hogy a parancs-file további feldolgozását befejezze-e.

Ha Y-t adunk meg, akkor a parancs file további parancsait átlépi, figyelmen kívül hagyja, és a DOS prompt jelenik meg. Ha N-t adunk meg, akkor áttér a parancs-file következő parancsainak feldolgozására. Ha bármilyen egyéb karakter (sorozat) a válasz, akkor a DOS megismétli az üzenetet.

Megjegyzés:

1. A parancs-file-ok tetszőleges szerkesztő programmal (pl. EDLIN) vagy a COPY CON paranccsal hozhatók létre.
2. A parancs-file-t tartalmazó lemezt a végrehajtás során nem lehet kivenni. Ha ez megtörténik, a DOS üzenetet küld, melyben kéri a felhasználót, hogy a lemezt helyezze vissza a meghajtóba.
3. A parancs-file utolsó sora (rekordja) egy másik parancs-file azonosítója is lehet, így közvetlenül hívható a következő parancs-file.
4. A parancs-file végrehajtása során az aktuális tartalomjegyzék tetszés szerint változtatható, mivel a rendszer feljegyzi, hogy a file melyik tartalomjegyzékben található.
5. A DOS 3.30 verziója alatt már lehetőség van egyik parancsfile-ból másikat úgy aktivizálni, hogy annak lefutása után a parancsok feldolgozása a hívó parancs-file-ban (az aktivizáló parancs után) folytatódik.

7.1 A parancs-file-ok meghívása és paraméterezése

A parancs-file meghívásának, azaz a benne foglalt parancsok és alparancsok végrehajtatásának általános alakja:

[*d:*][*út*]*file-név*[.BAT] [*paraméterek*]

Látható, hogy a kiterjesztés kötelezően .BAT.

7.1.1 Formális paraméterek

A file-ban a %0-%9 azonosítóval jelzett formális paraméterek adhatók meg. A %0 fenntartott paraméter, mivel ennek az értéke az eljárás végrehajtása alatt a parancs-file meghajtójának azonosítója (ha ez meg volt adva), és a parancs-file neve. A formális paraméterek az eljárásban a meghajtók és file-ok azonosítói helyén fordulhatnak elő.

Példa:

Legyen az ASMCOPY.BAT parancs-file-ban leírt eljárás:

```
COPY %1.MAC %2.MAC
COMP %1.MAC %2.MAC
TYPE %1.MAC
```

Ha az eljárást a következő módon indítjuk:

```
ASMCOPY A:FIRST1 B:FIRST2
```

akkor a következő parancsok hajtódnak végre:

```
COPY A:FIRST1.MAC B:FIRST2.MAC
COMP A:FIRST1.MAC B:FIRST2.MAC
TYPE A:FIRST1.MAC
```

A paraméterek értéke a hívás után a következő volt:

%0 - mindig a meghajtó- és a parancs-file azonosítóját tartalmazza. (Nem kell megadni)
%1 - a FIRST1 karaktereket tartalmazza
%2 - a FIRST2 karaktereket tartalmazza

Megjegyzés:

Abban az esetben, ha a parancs-file-ban egy olyan file-azonosító szerepel, amelynek a százalék (%) karakter része, akkor a százalékot kettőzve kell megadni.

Például:

Az ASM%MAC.MAC egy megengedett azonosító. Ha ezt egy parancsfile-ban akarjuk megadni, akkor ehelyett az ASM%%MAC.MAC azonosítót kell beírni.

7.1.2 Környezeti változók használata

A parancs-file-ban a SET segítségével beállítható/lekérdezhető környezeti változók is használhatók. Ezek értékére nevüknek % jelek közé zárásával hivatkozhatunk:

%prompt%

prompt	tetszőleges, a SET parancsban előfordulható környezeti változó neve. Például a PATH környezeti változóra az alábbi formában hivatkozhatunk: **%PATH%**

Példa:

```
IF %cel% = = c: GOTO :fix
ECHO Tegyen be egy lemezt!
PAUSE
:fix
program > %cel%mentes.txt
```

7.2 Az AUTOEXEC.BAT file

A DOS indításakor azon a lemezen, ahonnan indítottuk, a parancsfeldolgozó program megkeresi a főtartalomjegyzékben az AUTOEXEC.BAT file-t, és automatikusan végre is hajtja. Ide betehetjük azokat a parancsokat, amelyeket a DOS minden indításakor és újraindításakor végre akarunk hajtatni.

Például:

A >COPY CON AUTOEXEC.BAT < ENTER >	a file létrehozása
PROMPT pg < ENTER >	a DOS prompt ennek hatására az aktuális megbajtó betűjele lesz, a > karakterrel követve (ld. PROMPT parancs)
PATH \ < ENTER >	beállítja a főtartalomjegyzéket (ld. PATH parancs)
DATE < ENTER >	bekérjük a dátumot
TIME < ENTER >	és időt
< F6 >	Ctrl + Z: file-vége jel
< ENTER >	

Ha a parancs-file írásakor hibázunk, adjunk < Ctrl-Break >-et. Ez a COPY-t anélkül fejezi be, hogy a file létrejönne. Ekkor újra kezdhetjük a file létrehozását.

Az egyes parancsokat < ENTER >-rel, a file írását az < F6 > majd < ENTER > gombokkal fejezzük be. Ennek hatására a COPY kiüzeni:

```
1 File(s) copied
```

Amikor gépünkön dolgozni kezdünk, nézzük meg, van-e már AUTOEXEC.BAT file (pl. a DIR paranccsal), ha nincs, a fenti módon hozzuk létre. Ha van, listázzuk ki (TYPE parancs), és ha a fenti négy parancsot nem tartalmazná, érdemes ezeket is megadni.

7.3 DOS alparancsok

Az ebben a szakaszban ismertetésre kerülő alparancsok csak parancs-file-okban szerepelhetnek. Alkalmazásukkal a parancs-file-ok feldolgozását lehet vezérelni.

Például:

- A parancs-file-ban parancsok átlépése a végrehajtás során;

- Egy parancs többszöri végrehajtása;
- A DOS parancsok kijelzése a képernyőre a végrehajtás során;
- Üzenetek küldése az operátornak.

Valamennyi alparancs típusa belső.

7.3.1 REM - megjegyzés

Ennek az alparancsnak a segítségével a parancs-file-ban megjegyzést lehet elhelyezni. A megjegyzés tetszőleges karaktersorozat lehet, amely emlékeztető, figyelmeztető stb. utalás lehet egy későbbi felhasználás során. Az alparancsot megjegyzés nélkül is lehet használni. A REM alparancs a parancs-file tagolását segíti.

A parancs formája:

REM [*megjegyzés*]

7.3.2 ECHO - parancsok megjelenítése

Ez az alparancs lehetővé teszi, hogy a parancs-file parancsainak a képernyőn való megjelenítése felől döntsünk. Az ECHO a rendszerüzenetek kiírását nem befolyásolja.

A parancs formája:

ECHO [ON | OFF | *üzenet*]

A paraméter jelentése:

A rendszer indítása után a parancs-file-ok végrehajtásra kerülő parancsai a képernyőn megjelennek, azaz az ECHO ON (bekapcsolt) állapotban van.

Az ECHO OFF segítségével ez az üzemmód megváltoztatható, ekkor egyik parancs sem jelenik meg a képernyőn.

Ha az ECHO "üzenet" paraméterrel szerepel a parancs-file-ban, akkor az "üzenet" mindig megjelenik a képernyőn, függetlenül attól, hogy az ECHO ON vagy OFF állapotban van-e.

Ha az ECHO parancsait paraméterek nélkül adjuk meg, akkor pillanatnyi értéke jelenik meg: ON vagy OFF.

Példa:

Egy parancs-file részletét mutatjuk be az ECHO alparancs alkalmazására:

Legyen a PARANCSF.BAT azonosítójú parancs-file tartalma:

```
ECHO BEGIN PF
ECHO OFF
DIR C:AUTO*
```

```
ECHO ON
DIR C:AUTO*
ECHO ENDECHO
```

Ekkor a PARANCSF utasítás hatására a képernyőn a következő jelenik meg:

```
C:\>echo begin pf
begin pf
C:\>echo off
```

A DIR parancs kiírását az ECHO OFF állapot sem tiltja le, de maga a parancs nem látható. A file-ok kiírása után még a következő üzenetet kapjuk:

```
C:\>echo endecho
endecho
```

Megjegyzés:

Ha a parancs-file kiírásait úgy akarjuk szeparálni, hogy üres sorok is legyenek, akkor nem használhatjuk a paraméter nélküli ECHO parancsot (az ui. az aktuális értéket adja vissza), hanem a sorba valamilyen nem látható (pl. 255 ASCII kód) karaktert kell helyezni. Ha a parancs file-ból a felhasználót hangjelzéssel is figyelmeztetni akarjuk, az ECHO üzenet sorába a 7 ASCII kódot kell beírni.

(Minden ASCII kód beadható a numerikus billentyűzetről az <Alt> billentyű lenyomása közben decimálisan beírva az értéket.)

7.3.3 PAUSE - feldolgozás megszakítása

Ez az alparancs lehetővé teszi, hogy felfüggesszük a rendszer működését. Erre akkor van szükség, ha valamilyen operátori tevékenységet, pl. lemezcserét meg kell a rendszernek várnia. Ha a tevékenységet elvégeztük, akkor egy tetszőleges billentyű lenyomásával a rendszer továbbindítható.

A PAUSE segítségével egy parancs-file PAUSE utáni részének további feldolgozása felől is dönthetünk. Mivel hatására a DOS megáll, meggondolhatjuk, szükség van-e a parancs további részének végrehajtására.

A parancs formája:

PAUSE [*üzenet*]

A paraméter jelentése:

Az *üzenet* legfeljebb 121 karakter. Ha a vezérlés a PAUSE-ra kerül akár parancs-file-ban van a PAUSE, akár közvetlen utasításként adtuk ki, kiírja először az "üzenet"-et, majd kiírja:

Strike any key, when ready...

Ez azt jelenti, hogy ha készen vagyunk (a gondolkodással vagy egyéb tevékenységgel), nyomjunk le egy tetszőleges billentyűt.

Ha a PAUSE a parancs-file egy utasítása, és nem akarjuk a parancs-file további részét végrehajtani, adjunk <Ctrl-Break>-et. Ekkor megkérdezi:

Terminate batch job (Y/N)?

Azaz, befejezze-e a parancs-file végrehajtását. Ha Y-t adunk, befejezi, ha N-et, akkor nem. Egyéb karakter beírása esetén a kérdés megismétlődik.

Megjegyzés:

Természetesen az ECHO OFF utasítással megakadályozhatjuk az "üzenet" kiírását, de a "Strike any key..." és egyéb rendszer-üzenet továbbra is látható.

Példa:

A következő parancs-file részletben az operátort a nyomtató beállítására kérjük fel:

```
ECHO ON
PAUSE T. Operátor, ellenőrizd a nyomtatóban a papír beállítását. Ha kész, nyomj
meg egy tetszőleges billentyűt!
REM File-ok kiíratása a nyomtatón.
TYPE A:%2 >PRN
```

7.3.4 IF - feltételes végrehajtás

Ez az alparancs lehetővé teszi a parancs-file-beli DOS parancsok feltételes végrehajtását. A feltételek a következők lehetnek:

- az előző program befejezési kódjának értéke,
- két karaktersorozat összehasonlításának eredménye,
- megadott file létezése.

A parancs formája:

IF [NOT] *feltétel parancs*

A paraméterek jelentése:

A *feltétel* a következők egyike lehet:

- **ERRORLEVEL** *érték*
- *karaktersorozat1* = = *karaktersorozat2*
- **EXIST** *file-azonosító*

Ha a feltétel IGAZ, akkor a feltétel mögött álló DOS parancs végrehajtásra kerül (általában ez egy GOTO alparancs).

Az **ERRORLEVEL** *érték* feltétel akkor IGAZ, ha az előző program az "*érték*"-kel egyenlő, vagy annál nagyobb befejezési kódot (exit kód) állított be. Az érték decimális szám.

A *karaktersorozat1* == *karaktersorozat2* feltétel akkor IGAZ, ha a két sorozat minden pozícióban azonos jelet tartalmaz (egy nagybetűt nem tekint a rendszer a megfelelő kisbetűvel azonosnak).

Az **EXIST** *file-azonosító* feltétel akkor IGAZ, ha a rendszer a megadott file-t eléri. Megengedettek a globális név-karakterek (? és *).

A NOT a feltétel tagadását jelenti.

Példa:

1. A példában egy file létezését vizsgáljuk.

```
IF EXIST %1 GOTO G1
.
.
.
:G1
REM a file létezik, kiíratása következik
TYPE %1
```

2. A példában egy olyan program befejezését ellenőriztük, amelynél a sikeres befejezés esetén a befejezési kód = 0, sikertelen befejezés esetén pedig 1.

```
PROGRAM1
IF ERRORLEVEL 1 GOTO G1
PAUSE a program sikeresen befejeződött, cseréld le a lemezt az A: meghajtón.
GOTO G2
:G1
ECHO a PROGRAM1 hibás!
.
.
:G2 ...
```

3. A példában két karaktersorozat egyezését vizsgáljuk.

```
IF %1 = = A GOTO G2
.
.
:G2
REM az A: meghajtóra történik a mentés.
PAUSE a rendszer mentés lemezét tedd az A: meghajtóba. Ha kész, nyomj le egy
tetszőleges billentyűt.
BACKUP C: A:/M
REM mentésre kerültek a fix lemezről azok a file-ok, amelyek az utolsó mentés óta
változtak.
```

7.3.5 GOTO - ugrás a parancs-file-ban

A parancs-file-ban a parancsok végrehajtása a file-beli sorrendjükben történik. Ezt a sorrendet lehet a GOTO alparanccsal megváltoztatni.

A parancs formája:

GOTO *cimke*

A paraméter jelentése:

A *cimke* egy karakterekből álló jelsorozat. A cimkéket első nyolc karakterük különbözteti meg. A végrehajtás a parancs-file-ban a cimke utáni első DOS paranccsal folytatódik. Ha a végrehajtás sorrendjének megfelelően jut a vezérlés egy címkére, akkor a címke hatástalan.

Megjegyzés:

1. Ha a GOTO után írt címke nem található a file-ban, akkor üzenetet kapunk (label not found) és a végrehajtás befejeződik.
2. Az ugrások végrehajtásakor a címkék nem kerülnek kijelzésre. A címke után egy tetszőleges megjegyzés (lásd REM alparancs) írható. Ez segíti a parancs-file ellenőrzését, áttekintését.
3. A GOTO alparancs a parancs-file struktúráját rontja. Nem körültekintő alkalmazása esetén könnyen előáll a végtelen ciklus.
4. A *cimke*-t a file-ban bárhol elhelyezhetjük, a vezérlés az azt követő első parancsra adódik. A *cimke*-t tartalmazó sorban más nem lehet, s a *cimke* elé kettőspontot (:) kell rakni.

Példa:

Példánkban egy egyszerű végtelen ciklust mutatunk be.

```
:CIKLUS
ECHO ez egy végtelen ciklus!
GOTO CIKLUS
```

7.3.6 FOR - DOS parancsok ismételt végrehajtása

Ez az alparancs lehetővé teszi, hogy egy DOS parancsot többször végrehajtson a rendszer anélkül, hogy a parancsot ismételten megadnánk.

A parancs formája:

FOR *változó* **IN** (*mondat*) **DO** *parancs*

A paraméterek jelentése:

A "*változó*" egyenként felveszi a "*mondat*" egyes elemeinek értékét, és ezzel az értékkel a DO utáni "*parancs*" rendre végrehajtódik, amennyiben a parancs ezzel az értékkel értelmes. A "*változó*" formája:

%karakter

ahol *karakter* egy betű vagy egy számjegy, ha a FOR-t DOS utasításként, parancs-file-on kívül használjuk, és

%%karakter,

ahol *karakter* a fenti, ha a FOR parancs-file-beli utasítás.

A "*mondat*" legalább egy, (egymástól szóközökkel elválasztott) elemből áll. Az egyes elemekkel együtt a "*parancs*"-nak értelmesnek kell lennie.

A "*parancs*" az ismétlődő DOS parancs, paramétereivel együtt. Általában paramétereiben szerepel a "*változó*".

Megjegyzés:

1. A mondat elemeiben, ha azok file-nevek, a globális file-név jelek is használhatók.
2. A FOR alparancsban út nem adható meg.
3. A FOR alparancsok nem skatulyázhatók egymásba, azaz a DO-t nem követheti újabb FOR (ha megkíséreljük, kiírja: "FOR cannot be nested").

Példa:

1. A következő parancs szerepel a parancs-file-ban:

```
FOR %%IN (PROG1.ASM PROG2.ASM PROG3.ASM) DO
COPY A:%%IN B:
```

Ez a következő parancsokkal egyenértékű:

```
COPY A:PROG1.ASM B:
COPY A:PROG2.ASM B:
COPY A:PROG3.ASM B:
```

2. Próbáljuk ki a DOS prompt után a következő parancsot:

```
FOR %x IN (AUTOEXEC.BAT COMMAND.COM) DIR %x
```

Hatása a két file-ra külön-külön kiadott DIR paranccsal egyenértékű.

7.3.7 SHIFT - a paraméterek léptetése

Ez az alparancs lehetővé teszi, hogy a felhasználó tíznél több paramétert használjon (%0-%9 voltak a megengedett formális paraméterazonosítók).

A parancs formája:

```
SHIFT
```

A parancs működése

A formális paraméterek száma tíz. Ebből a %0 paraméter aktuális értéke mindig a meghajtó és a parancs-file-azonosító.

Ha a felhasználónak tíznél több paraétert kell használni, akkor SHIFT alparancsot alkalmazhat. Az alparancs hatására a parancssor minden paramétere eggyel balra tolódik, tehát a %0 új értéke a %1 régi értéke, %1 új értéke a %2 régi értéke stb.

Az alparancs akkor is használható, ha egy másik parancs-file-t hívunk meg a file utolsó utasításában.

Példa:

Készítsük el a következő PARANCSF.BAT parancsfile-t:

```
ECHO OFF
ECHO PARF KEZDET
TYPE %0.BAT
FOR %%X IN (%1 %2 %3 %4 %5 %6 %7 %8) DO TYPE %%X
PAUSE FOR UTAN a PAUSE utasításokat azért tesszük be, hogy egy képernyő
teleírása után megálljon a rendszer
COPY %9 %8
TYPE %8
PAUSE SHIFT ELOTT
SHIFT
TYPE %9
ECHO PARF VEGE
```

Legyen a PR1, PR2, PR3, PR4, PR5, PR6, PR7, PR8, PR9 és PR10 file-unkban csak ez az egy sor:

```
EZ PRn
```

ahol n a megfelelő file nevében szereplő szám. Ezután hívjuk meg parancs-file-unkat a következő módon:

```
PARANCSF PR1 PR2 PR3 PR4 PR5 PR6 PR7 PR8 PR9 PR10
```

Ennek eredményeképpen megkapjuk először magának a parancs-file-nak, majd a többi file-unknak a listáját, összesen 11 db listát. Megjegyezzük, hogy a COPY végrehajtása után természetesen a PR8 tartalma:

```
EZ PR9
```

A többi file tartalma nem változik.

7.3.8 CALL - parancs-file hívása

A CALL alparancs segítségével a parancs-file-ban egy másik parancs-file végrehajtását rendelhetjük el, hasonlóan a szubrutin-híváshoz.

A parancs formája:

CALL [*d:*][*út*]*file-név*

8. A sorszerkesztő program (EDLIN)

A számítógép használata során gyakran lesz arra szükségünk, hogy ún. szöveges (text) file-okat állítsunk elő és tároljunk lemezen. Ilyenek például:

- magasszintű nyelvek (pl. PASCAL, FORTRAN) vagy az assembly nyelv forrásprogramjai,
- olvasásra, tájékoztatásra szánt szövegek (levelek, dokumentációk),
- számítógépes feldolgozásra kerülő file-ok (csak karakteres ábrázolású adatokkal),
- DOS parancs-file-ok.

Eddigi ismereteink szerint a COPY paranccsal tudunk egy szöveget a billentyűzetről a lemezre másolni, például így:

```
COPY CON MASOL.BAT< ENTER >
DISKCOPY A: B:< ENTER >
DISKCOMP A: B:< F6 > - < ENTER >
```

A fenti példában egy kétsoros parancs-file-t hoztunk létre, amelyet a MASOL paranccsal lehet behívni. Ez az eljárás egyszerűnek tűnik, ha a szöveg beírása közben hibát tartalmazó sort még nem zártuk le az < ENTER >-rel, mert visszaléphetünk a < Backspace > (vissz-szatörlés) billentyűvel és javíthatunk. Ha azonban a hibás sort már elhagytuk, nincs lehetőségünk a korrigálásra, és újra be kell írnunk a teljes szöveget. A lemezen tárolt file-ok módosítására pedig a COPY utasítás végképp alkalmatlan.

Szerencsére azonban más módszer is kínálkozik a szöveges file-ok létrehozására és módosítására. Erre a célra speciális programok (editorok, szövegszerkesztők) állnak rendelkezésre, egyet közülük a DOS rendszerlemezen is megtalálunk, EDLIN néven. (A PROPOS-16 operációs rendszerben a hasonló szerkesztő program neve: EDIT.)

Az EDLIN (Line Editor, sorszerkesztő) program az alábbi feladatok végrehajtására képes:

- új szöveges file-ok létrehozása és tárolása lemezen.
- meglevő szöveges file-ok módosítása, az új és a régi változat megőrzésével,
- a szöveg egyes sorainak kiíratása, módosítása, törlése, másolása,áthelyezése, új sorok beszúrása,
- szövegrészek keresése, helyettesítése vagy törlése.

A felsorolásból látszik, hogy gyakorlatilag minden műveletet el lehet végezni a szövegeken az EDLIN segítségével. Más kérdés viszont, hogy mennyire kényelmes és hatékony ez a munka. Az EDLIN - a nevéből következően - csak sorszerkesztő, ezért az egyes sorok írása, módosítása közben nem léphetünk át másik sorba, csak ha előbb az adott sor szerkesztését befejeztük, < ENTER >-rel lezártuk. A művelet végrehajtásához külön parancsokat kell kiadni, és nem lehet a szövegben felfelé vagy lefelé lapozni (scroll-ozni). Aki a BASIC teljes képernyős editorához szokott, nehezen tudja elfogadni az EDLIN korlátait.

Ha hosszabb programokat, szövegeket kell írnunk, célszerű tehát az EDLIN helyett egy szövegszerkesztőt (például a WordStart) vagy egy teljes képernyős editort (például a Personal Editort) alkalmazni. Kisebb módosításokra vagy rövid szövegek írására viszont az EDLIN tökéletesen megfelel, kis mérete, rövid betöltési ideje, egyszerű és hatékony használata is előnyt jelent.

Az EDLIN által létrehozott szöveges file sorokra tagolódik, az egyes sorok váltakozó hosszúságúak lehetnek, de legfeljebb 254 karaktert tartalmazhatnak. A sorokat CR-LF (kocsivissza-soremelés) jelek zárják le. A sorok számát csak a file-t tartalmazó lemez kapacitása korlátozza; az EDLIN-nel az operatív memória méreténél hosszabb szöveget is fel lehet dolgozni.

Az ún. bináris file-ok (pl. az object- és a végrehajtható programokat tartalmazó file-ok) módosítására az EDLIN nem alkalmas. Erre a célra a DEBUG rendszerprogramot ajánljuk, melyet a 9.fejezetben ismertetünk.

8.1 Az EDLIN indítása

Az EDLIN program a DOS rendszerlemezen található, EDLIN.COM néven, de szükség szerint átmásolhatjuk a szöveges file-t tartalmazó lemezre is. Külső program lévén fontos, hogy az EDLIN indításakor a programot tartalmazó lemez valamelyik meghajtóegységben legyen.

Az EDLIN indító parancsának általános formája:

EDLIN [*d:*][*út*]*file-név*[*.kit*][/**B**]

ahol [d:][út]file-név[.kit] a létrehozandó, illetve módosítandó file specifikációja.

/B-t akkor kell megadni, ha a file-on belül file-vége(<Ctrl-Z>) karakterek találhatók. (/B nélkül ugyanis az EDLIN az első file-vége jelnél abbahagyja a file beolvasását.)

Ha egy új szöveges file-t akarunk létrehozni, akkor az indító parancsban a lemezen nem szereplő file nevét kell megadnunk, például:

```
A>B:EDLIN SZOVEG.TXT
New file
*
```

A példában az EDLIN programot tartalmazó lemez a B egységben található, az új file-t (SZOVEG.TXT) pedig az aktuális meghajtón, az A egységen hozzuk létre. Az EDLIN a bejelentkezésekor kiírta: "New file" (új file), majd megjelent az EDLIN prompt-ja (felszólító jele), a csillag, amely azt jelenti hogy a program a parancsunkat várja. Az EDLIN a fenti indítás hatására megnyitja a SZOVEG.$$$ nevű ideiglenes file-t, amellyel a feldolgozás végén két dolog történhet. Ha tárolással fejezzük be a szerkesztést ("E" parancs), akkor a SZOVEG.$$$ megkapja a ténylegesen előírt kiterjesztést, és SZOVEG.TXT lesz belőle. Ha tárolás nélkül lépünk ki az EDLIN-ből ("Q" parancs), akkor pedig a SZOVEG.$$$ törlődik a lemezről.

Ha az EDLIN-nel egy már meglevő szöveges file-t akarunk módosítani, akkor az indító parancsba a megfelelő file nevét kell írnunk. Ebben az esetben szükséges lehet a /B opció használata is. Egy példa:

```
A>EDLIN B:MEGLEVO.DOC
End of input file
*
```

A programot az A egységről indítottuk, a MEGLEVO.DOC nevű file-t tartalmazó lemezt pedig a parancs kiadása előtt behelyeztük a B egységbe. Az "End of input file" (input file vége) üzenet azt jelzi, hogy az EDLIN a file teljes szövegét beolvasta a memóriába. Ha a szöveg nem fér el a rendelkezésre álló szabad memóriában, akkor az "End of input file" szöveg nem jelenik meg, csak az EDLIN prompt-ja, a csillag. Megjegyezzük, hogy az EDLIN csak egy 64kB-os memóriaterületet tud felhasználni, ezért a hosszabb szövegeket a nagy memóriakapacitású gépeken is csak részletenként lehet feldolgozni.

Meglévő file módosításakor az EDLIN az adatok biztonsága érdekében a következőképpen jár el: a file új, módosított változata számára először egy .$$$ kiterjedésű file-t nyit meg (példánkban a MEGLEVO.$$$-t), de az eredeti file-hoz nem nyúl. Ha a szerkesztést tárolással fejezzük be, akkor a régi eredeti file a .BAK kiterjesztést kapja (MEGLEVO.BAK), és a lemezen marad. A BAK rövidítés a backup-ra, biztonsági másolatra utal. Ha már volt a lemezen egy korábbi MEGLEVO.BAK file, akkor az az átnevezés előtt törlődik. Ezután az új, módosított file megkapja a végleges nevét (MEGLEVO.$$$ helyett a MEGLEVO.DOC nevet)

Ha a szerkesztésből tárolás nélkül lépünk ki (pl, ha hibásan hajtottuk végre a módosítást és újra akarjuk kezdeni az eredeti file-lal) akkor az EDLIN egyszerűen törli a .$$$ kiterjesztésű ideiglenes file-t és - mivel a régi file megmaradt- ezzel vissza is áll az eredeti állapot.

Az EDLIN ismertetett file-kezelési stratégiája következtében a módosított szövegek (például a fejlesztés alatt álló forrásprogramok) mindig két generációban állnak rendelkezésre: az egyik a legfrissebb, a másik az eggyel korábbi változat. E módszernek két további következménye: a .BAK kiterjesztésű file-okat nem lehet az EDLIN-nel feldolgozni (ebben az esetben megegyezne az eredeti és az archív file neve), ezért ilyenkor az illető file-t más névre kell átneveznünk. A másik következmény: a file-ok módosításakor az eredeti és az archív változat egyazon lemezen helyezkedik el, ezért együttes méretük nem haladhatja meg a lemezen rendelkezésre álló területet.

Egy példa a LEVEL.BAK nevű szöveg átszerkesztésére:

```
A>RENAME B:LEVEL.BAK LEVEL.TXT
A>EDLIN B:LEVEL.TXT
End of input file
*
```

A szerkesztés tárolással történt befejezése után a módosított szöveg a LEVEL.TXT-be, a régi szöveg pedig a LEVEL.BAK file-ba kerül.

8.2 Az EDLIN parancsai

Az EDLIN elindítása után egy csillag jelzi a képernyőn, hogy a program a parancsainkat várja. Ilyenkor EDLIN utasításokat kell beírnunk, amelyeknek semmi köze nincs a DOS parancsaihoz. E speciális parancsokkal majd sorban megismerkedünk, de először nézzük meg a parancsok néhány általános elemét.

Az EDLIN sorszerkesztő, ezért a legtöbb parancsában utalnunk kell arra, hogy az adott műveletet mely soron vagy sorokon kell végrehajtania. Az EDLIN a sorokat 1-től kezdve, egyesével számozza. A sorszámok nem részei a soroknak, nem tárolódnak a lemezen, de a képernyőn megjelennek a tájékozódás végett. Egy szövegrész például így íródhat ki:

```
1: Ez az elso sor
2: ez pedig a masodik
3:*Ez az aktualis sor
4: Ez itt a 4. sor
```

(Az ékezetes betűket természetesen nem tudjuk beírni, de így olvashatóbb lesz a szöveg.)

A példában tehát az 1-től 4-ig terjedő sorszámok (valamint a kettőspontok és a csillag) nem részei a szövegnek, de általuk tudjuk meg, hogy az egyes sorszámokra milyen számmal lehet hivatkozni. A harmadik sor szövege előtt látható csillag (ne tévesszük össze az EDLIN prompt-jával) azt jelzi, hogy ez az ún. aktuális sor.

A parancsokban a sorokra általában sorszámokkal hivatkozhatunk (1-től 65529-ig), de két speciális jel is megadható a sorszámok helyett. Ezek jelentése a következő:

. (pont) az aktuális sort jelzi
(hashmark) az utolsó utáni sort jelzi

Abszolút sorszámok helyett az aktuális sorhoz viszonyított relatív hivatkozásokat is használhatunk. Ha egy parancsban +10-et (ill. -10-et) írunk, akkor ezzel az aktuális sornál 10-zel nagyobb (illetve 10-zel kisebb) sorszámra hivatkozunk.

A legtöbb EDLIN parancs egyetlen betűből áll, amely előtt és/vagy után paraméterek szerepelhetnek. A parancsokat kis vagy nagy betűvel is írhatjuk. Kivételt egyedül a "sor szerkesztése" parancs képez, amelynek nincs betűjele, csak az átszerkesztendő sor számából áll.

A numerikus paramétereket vesszővel vagy szóközzel választjuk el egymástól. Egy sorba több EDLIN parancsot is írhatunk, ilyenkor ezeket pontosvesszővel kell egymástól elválasztanunk. Ha a pontosvessző előtt egy "keresés" vagy "helyettesítés" parancs áll, akkor azok utolsó string-adatát <ENTER> helyett <F6>-tal kell lezárni. A parancsokat az EDLIN csak az <ENTER> billentyű lenyomásakor hajtja végre.

Ha a szövegbe a <Ctrl> billentyűvel képzett speciális karaktereket akarunk írni, akkor először a <Ctrl-V> billentyűket kell lenyomnunk, majd a kívánt karakter nagybetűs alakját. A <Ctrl-Z> kombinációt például így állíthatjuk elő:

<Ctrl-V> Z

Az EDLIN futtatása során a megszokott módon alkalmazhatjuk valamennyi DOS szerkesztő billentyűt (<F1>, <F2>, <F3>, <F4>, <F5>, <Del>, <Backspace>, <ESC>, kurzor jobbra és balra) a parancsok és a szövegek írásánál is. (E billentyűk szerepét a 2.4 fejezetben ismertettük.) A parancsok végrehajtását a <Ctrl>-<Break>

kombináció szakítja meg, és a <Ctrl-NumLock> <Shift-PrtSc>, <Ctrl-PrtSc> billentyűk is ugyanúgy működnek, mint a DOS-ban.

8.2.1 I (insert) - sorok beszúrása

Az I (insert, sorok beszúrása, beillesztése) parancs nélkülözhetetlen az új file-ok létrehozásánál, valamint a meglevő szövegek új sorokkal való bővítésénél is. A parancs hatására az EDLIN "beszúró üzemmódba" tér át, és az újonnan beírt sorokat a megadott számú sor elé illeszti be. A parancs általános formája:

[sorszám]I

Ha a sorszámot nem adjuk meg, akkor a beszúrás az aktuális (csillaggal jelzett) sor elé történik. Ha pedig a # jelet használjuk, vagy a létezőknél magasabb sorszámra hivatkozunk, akkor az új sorok az eddigi szöveg utolsó sora után kerülnek.

A beszúró üzemmódban tetszőleges számú új sort írhatunk be, az <ENTER>-rel lezárva. Az EDLIN mutatja a folyamatosan növekvő sorszámokat, amelyeket az új sorok kapnak. Ha nem akarunk több sort beszúrni, a <Ctrl-Break> billentyűkkel lépbetünk ki a beszúró módból.

A beillesztések elvégzése után az utolsó beszúrt sort követő sor lesz az aktuális sor (akkor is, ha a szöveg végéhez toldottunk hozzá új sorokat). Az aktuális sort, valamint az utána álló sorokat (ha vannak ilyenek) az EDLIN újraszámozza, így helyreáll az egyesével folyamatosan növekvő számozás rendje.

Első példánkban egy új file létrehozásához használjuk fel a beszúró programot:

```
A>EDLIN B:UJ.TXT
New file
*I <ENTER>
   1:* Tevedni emberi dolog, de <ENTER>
   2:* igazan osszekutyulni valamit <ENTER>
   3:* csak szamitogeppel lehet. <ENTER>
   4:*<Ctrl-Break>
*
```

Az I parancsot sorszám nélkül adtuk ki, de mivel a memória eddig üres volt, ezért az új sorok (természetesen) az első sortól kezdve illesztődnek a file-ba. Három sort írtunk be <ENTER>-rel lezárva, majd a negyedik sor helyett lenyomtuk a <Ctrl-Break>-et. Erre az EDLIN kilépett a beszúró módból, és visszatért parancs-szintre, amint azt a prompt (csillag) megjelenéséből látjuk.

A fenti példában egészen rövid sorokat írtunk be. Lehetőségünk van azonban hosszú, de legfeljebb 254 karakteres sorok létrehozására is. Bár a 72. karakter begépelésekor eljutunk a képernyő jobb szélére, nyugodtan írhatjuk tovább a szöveget. Ilyenkor a sor folytatását az EDLIN a képernyő következő sorába írja. Egy logikai sor tehát - a megjelenítés szempontjából - több, maximum négy fizikai sorra tagolódhat. A 254. karakter beírása után az EDLIN hangjelzést ad, és több karaktert nem fogad el.

Ha a sorok írása közben hibát vétettünk, legegyszerűbben a < Backspace > (visszatörlés) billentyűvel, majd a hibás rész újbóli begépelésével javíthatunk. Felhasználhatjuk továbbá a DOS szerkesztőbillentyűit is (< F1 >, < F3 >, < Ins >, < Del > stb.).

Folytassuk a példát úgy, hogy az első sor elé illesztünk be újabb sorokat:

```
*1I
 1:*A megbizhatosag
 2:*otodik torvenye
 3:*<Ctrl-Break>
*L
 4:*Tevedni emberi dolog, de
 5: igazan osszekutyulni valamit
 6: csak szamitogeppel lehet.
*
```

Végül megnézzük, hogyan lehet beszúrni egy új sort az eddigi 3. sor elé:

```
*3I
 3:*(Murphy torvenykonyvebol):
 4:*<Ctrl-Break>
*L
 1: A megbizhatosag
 2: otodik torvenye
 3: (Murphy torvenykonyvebol):
 4:*Tevedni emberi dolog, de
 5: igazan osszekutyulni valamit
 6: csak szamitogeppel lehet.
*
```

Figyeljük meg a példánkban, mikor melyik lett az aktuális sor.

8.2.2 L (list) - sorok listázása

Az L (list, listázás) parancs hatására az EDLIN megjeleníti a kijelölt sorokat a képernyőn. Eközben az aktuális sorszám nem változik. A parancs általános formája:

[sorszám1][,sorszám2]L

Ha mindkét sorszámot megadtuk, akkor a sorszám1-től a sorszám2-ig terjedő sorok jelennek meg (beleértve a két határértéket is). A sorszám1 nem lehet nagyobb a sorszám2-nél, különben "Entry error" (hibás parancs) üzenetet kapunk.

Ha mindkét sorszámot elhagyjuk, és csak az L kulcsszót írjuk be, akkor általában az aktuális sor előtti tizenegy sor, az aktuális sor és az azt követő tizenegy sor (azaz összesen huszonhárom sor) jelenik meg a képernyőn. Kivétel ez alól, ha az aktuális sor előtt nincs tizenegy sor, vagyis az aktuális sor száma tizenkettőnél kisebb. Ilyenkor az EDLIN az első huszonhárom sort jeleníti meg.

Ha az első sorszámot hagyjuk el és a parancsot így írjuk:

,sorszám2 L

akkor a kiíratás az aktuális sor előtti tizenegyedik sorral kezdődik és a kijelölt sorral végződik. Kivétel, ha a megadott sorszám eleve kisebb, mint az aktuális sor előtti tizenegyedik sor száma. Ilyenkor az EDLIN úgy hajtja végre a parancsot, mintha mindkét sorszámot elhagytuk volna.

Ha csak az első sorszámot írjuk ki:

sorszám1 L

akkor a megadott sorszámtól kezdve összesen huszonhárom sor jelenik meg.

A példában feltételezzük, hogy a szövegben ötven sor szerepel, és mindegyikben utalás található a sor sorszámára. (Ez az 1.sor, Ez a 2.sor stb). Az aktuális sor legyen a negyvenötödik.

Nézzük meg ezek után néhány listázási parancs hatását:

```
*L
 34: Ez a 34. sor
 35: Ez a 35. sor
 36: Ez a 36. sor
   .
   .
   .
 45:*Ez a 45. sor
 46: Ez a 46. sor
 47: Ez a 47. sor
 48: Ez a 48. sor
 49: Ez a 49. sor
 50: Ez az 50. sor
*11,14L
 11:Ez a 11. sor
 12:Ez a 12. sor
 13:Ez a 13. sor
 14:Ez a 14. sor
*,38L
 34:Ez a 34. sor
 35:Ez a 35. sor
 36:Ez a 36. sor
 37:Ez a 37. sor
 38:Ez a 38. sor
*14L
 14:Ez a 14. sor
 15:Ez a 15. sor
 16:Ez a 16. sor
```

```
       .
       .
  35:Ez a 35. sor
  36:Ez a 36. sor
*
```

Nézzük át a példákat, és ellenőrizzük a megadott szabályok szerint, hogy miért pont ezeket a sorokat írta ki az EDLIN. (Az első listázási parancsnál nem lehetett az aktuális sor után tizenegy sort megjeleníteni, mert az 50. sornál vége lett a szövegnek.)

8.2.3 P (page) - listázás laponként

A P (page, listázás laponként) parancs nagyon hasonlít az előző pontban ismertetett L (listázás) parancshoz, ez is a megadott sorok képernyőn megjelenítését szolgálja. A fő különbség abban rejlik, hogy a P parancs (az L-lel szemben) megváltoztatja az aktuális sorszámot (az utolsóként kiírt sor lesz az aktuális sor).

Különbözik továbbá az első sorszámra vonatkozó alapértelmezés is. A parancs általános alakja:

[sorszám1][,sorszám2]P

Ha mindkét sorszámot megadtuk, akkor a sorszám1-től a sorszám2-ig terjedő sorok jelennek meg (beleértve a két határértéket is). A sorszám1 nem lehet nagyobb a sorszám2-nél, különben "Entry error" (hibás parancs) üzenetet kapunk.

Ha elhagyjuk az első sorszámot, akkor a kiíratás az aktuális sor utáni sorral fog kezdődni. Ha a második sorszámot hagyjuk el, akkor az elsőként megadott sortól kezdve összesen huszonhárom sor jelenik meg. Ha mindkét sorszámot elhagyjuk, és csak a P parancsot írjuk ki, akkor mindkét sorszámra az előbbiekben ismertetett alapértelmezés érvényesül.

A P parancs (paraméterek nélküli) ismételgetésével tehát kényelmesen végiglapozhatunk, végignézhetünk egy file-t az elejétől a végéig.

A példában most is feltételezzük, hogy a szövegben ötven sor szerepel és mindegyikben utalás található a sor számára (Ez az 1. sor, Ez a 2.sor stb). Az aktuális sor legyen a negyvenötödik. Jelenítsük meg a szöveget a képernyőn az első sortól az utolsóig:

```
*1P
 1: Ez az 1. sor
 2: Ez a 2. sor
 3: Ez a 3. sor
      .
      .
      .
 22: Ez a 22. sor
 23:*Ez a 23. sor
*P
 24: Ez a 24. sor
 25: Ez a 25. sor
```

```
     .
     .
     .
  45: Ez a 45. sor
  46:*Ez a 46. sor
 *P
  47: Ez a 47. sor
  48: Ez a 48. sor
  49: Ez a 49. sor
  50:*Ez az50. sor
 *
```

Nézzük meg a példában, hogyan változik az aktuális sorszám.

8.2.4 Sorszám - sor szerkesztése

Ha egy meglevő file-t szerkesztünk át, vagy az I parancs kiadása után magunk írunk be egy szöveget, szükség lehet a memóriában elhelyezkedő sorok módosítására, javítására. Ezt a célt szolgálja a - némileg rendhagyó - "sor szerkesztése" parancs, melynek általános formája:

[sorszám]

A parancsnak nincs betűjele, csak az átszerkeszteni kívánt sor számát kell megadnunk. Sorszám helyett a "." (pont) is használható, amely az aktuális sorra utal. Ha elhagyjuk a sorszámot és csak az < ENTER > -t nyomjuk meg, akkor az aktuális sor utáni sort módosíthatjuk. Utóbbi módszerrel kényelmesen lehet az egymás utáni sorokat javítani.

A "sor szerkesztése" parancs hatására a kijelölt sor megjelenik a képernyőn. Az alatta levő sorban ismét kiíródik a sor száma, ebben a sorban lehet a módosítást vagy újraírást végrehajtani. Az alábbi példában egy program szövegének 24. sorát módosítjuk:

```
*24
 24:*DCL SUM  FIXED BIN (15);
 24:*
```

Az utoljára megjelenített üres sorba tetszésünk szerinti új szöveget is beírhatunk, amely a korábbi tartalomtól független. Az < ENTER > lenyomása után ez lesz a sor új tartalma és ez a sor lesz az új aktuális sor. Példa a teljes újraírásra:

```
*24
 24:*DCL SUM FIXED BIN (15);
 24:*DECLARE (X,Y) FLOAT DEC (5);
*
```

Előfordulhat az is, hogy csak kisebb változtatásokra van szükség, ilyenkor a sor régi tartalmát - az írásmunka csökkentése érdekében - felhasználjuk, a módosításokat pedig a DOS szerkesztő billentyűivel (<F1>, <F2>, <F3>, <F4>, <F5>, <Ins>, <Del>,

< Backspace >, < ESC >, kurzor jobbra és balra) végezzük el. (E billentyűk szerepét a 2.4 fejezetben ismertettük.) A következő példában a SUM változó nevét S-re javítjuk ki:

```
*24
 24:*DCL SUM FIXED BIN(15);
 24:*DCL S
```

A szerkesztő parancs kiadása után ötször lenyomjuk az < F1 > -et, erre megjelent a sor első öt karaktere. Ezután kétszer lenyomjuk a < Del > billentyűt, ami törli a következő két betűt ("UM"). Végül az < F3 > gombbal kiíratjuk a sor hátralevő karaktereit:

```
*24
 24:*DCL SUM FIXED BIN (15);
 24:*DCL S FIXED BIN (15);
*
```

A sorba új szövegrészeket az < Ins > billentyű lenyomása után lehet beírni (< Ins > nélkül az újonnan begépelt karakterek a sor régi tartalmát módosítják, felülírják). A szerkesztő billentyűk használatára további példákat találunk a 2.4 fejezetben.

A változtatások végrehajtása után, a sor végén, nyomjuk le az < ENTER > -t, ekkor lép a módosított sor az eredeti helyébe. Az < ENTER > -t csak a sor végén szabad lenyomni, különben a sor hátralevő, nem látható része elvész. Ezután az utolsóként módosított sor lesz az aktuális sor.

A szerkesztő parancs kiadása után még meggondolhatjuk magunkat és változtatás nélkül hagyhatjuk a kijelölt sort (például, ha rájövünk, hogy nem is ezt a sort akartuk módosítani). A szerkesztésből- módosítás nélkül - a következő módszerekkel lehet kilépni:

- Lenyomjuk az < ENTER > -t, amikor a kurzor a sor első karakterén áll. (Vigyázat, ha a kurzor nem a sor elejére mutat, akkor a módosító sorban látható szöveg lép az eredeti sor helyébe!)
- A sor szerkesztése közben bármikor lenyomjuk a < Ctrl-Break > billentyűket.
- Ha módosítás nélkül léptünk ki a szerkesztésből, akkor az aktuális sor száma sem változik meg.

A szerkesztő paranccsal csak az egyes sorok tartalmát tudjuk módosítani, új sorok beillesztésére az "I", sorok törlésére a "D" parancsot kell használni.

Az összefoglaló példában egy levél 9. sorát módosítjuk. Az eredeti szöveg ez volt:

```
9: Kedves Zsuzsa, 5-kor varlak a mozi elott.
```

Hirtelen változás folytán a "Zsuzsa" nevet "Évá"-ra, a találkozó helyét pedig "mozi" helyett "diszkó"-ra kell módositanunk. A javítás lépései:

```
*9
 9:*Kedves Zsuzsa, 5-kor varlak a mozi elott
 9:*Kedves
```

A változatlanul hagyott első szót az < F2 > billentyű, majd a nagy "Z" betű lenyomásával ismételtük meg (az eredeti szöveg másolása a "Z" első előfordulásáig). Ugyanez lett volna a hatás az < F1 > hétszeri lenyomásával is.

Ezután beírjuk az "Éva" nevet, és háromszor lenyomjuk a < Del > billentyűt (három betűvel hosszabb a "Zsuzsa" név). Ezt látjuk a képernyőn:

```
9:*Kedves Zsuzsa, 5-kor varlak a mozi elott.
9:*Kedves Eva
```

Nyomogassuk az < F1 > billentyűt mindaddig, amíg a régi szöveget változtatás nélkül akarjuk átvenni. Tizenhét lenyomással tudjuk az ", 5-kor várlak a" szavakat megismételni:

```
9:*Kedves Zsuzsa, 5-kor varlak a mozi elott.
9:*Kedves Eva, 5-kor varlak a
```

Mivel a "diszkó" szó két betűvel hosszabb a "mozi"-nál, az első két karakter számára az < Ins > billentyűvel csinálunk helyet. Lenyomjuk az < Ins > gombot, majd a "d" és "i" betűket, végül az < Ins > újbóli lenyomásával lépünk ki a beszúró módból:

```
9:*Kedves Zsuzsa, 5-kor varlak a mozi elott
9:*Kedves Eva, 5-kor varlak a di
```

A "mozi" szó karakterei helyére egyszerűen begépeljük a "szkó" betűsorozatot, majd az < F3 > lenyomásával átmásoltatjuk a sor háralevő részét (az "előtt" szót). Ezzel befejeztük a változtatásokat és eljutottunk a sor végére. Most már lenyomhatjuk az < ENTER >-t:

```
9:*Kedves Zsuzsa, 5-kor varlak a mozi elott.
9:*Kedves Eva, 5-kor varlak a diszko elott.
*
```

8.2.5 D (delete)- sorok törlése

Most már tudjuk, hogyan kell új sorokat létrehozni és hogy lehet a meglevő sorokat módosítani. Ismerkedjünk meg ezek után a D (delete, törlés) paranccsal, melynek segítségével a feleslegessé vált sorokat teljes egészében kitörölhetjük.

A parancs általános formája:

[sorszám1][,sorszám2]D

Ha mindkét sorszámot megadjuk, akkor a sorszám1-től a sorszám2-ig terjedő sorok törlődnek (beleértve a két határértéket is.)

Ha elhagyjuk a második sorszámot, és a parancsot így írjuk:

sorszám1 D

akkor csak a kijelölt sor törlődik.

Ha csak a második sorszámot írjuk ki:

,sorszám2 D

akkor az aktuális sortól kezdve a megadott számú sorig terjedő szövegrészt törli az EDLIN.

Ha mindkét sorszámot elhagyjuk, és csak a D kulcsszót írjuk ki, akkor csak az aktuális sor törlődik.

A parancs végrehajtása után az utolsó kitörölt sort követő sor lesz az aktuális sor (àkkor is ha az utolsó sort töröltük ki). Az aktuális sort, valamint az utánna álló sorokat az EDLIN újraszámozza, így helyreáll a folyamatos sorszámozás rendje.

A törlési parancsot a következő szövegen fogjuk gyakorolni:

```
 1: Az EDLIN hibauzeneteinek listaja:
 2:-------------------------------
 3: Cannot edit .BAK file -rename file
 4: (BAK file-t nem lehet szerkeszteni, nevezze at)
 5: Disk full - write not completed
 6: (A lemez betelt - a tarolas nem fejezodott be)
 7: Entry error
 8: (Hibas parancs)
 9: Line too long
10: (Tul hosszu sor)
11: Must specify destination line number
12: (meg kell adni a cel-sor szamat)
13:*No room in directory for file
14: (Betelt a directory /tartalomjegyzek/ a lemezen)
15: Not enough room to merge the entire file
16: (Keves a memoria a teljes file beolvasasahoz)
17: Not found
18: (Nem talalhato)
```

Töröljük a 9-től 14-ig terjedő sorokat:

```
*9,14D
*L
 1: Az EDLIN hibauzeneteinek listaja:
 2:---------------------------
 3: Cannot edit .BAK file -rename file
 4: (BAK file-t nem lehet szerkeszteni, nevezze at)
 5: Disk full - write not completed
 6: (A lemez betelt - a tarolas nem fejezodott be)
 7; Entry error
 8: (Hibas parancs)
 9:*Not enough room to merge the entire file
10: (Keves a memoria a teljes file beolvasasahoz)
11: Not found
12: (Nem talalhato)
*
```

A 15-18. sorokat újraszámozza az EDLIN (9-12. lett a számuk). A 9.sor lett az aktuális sor.

Ha most a 9-10. sorokat akarjuk törölni, a D parancsban elhagyhatjuk az első sorszámot, hiszen az alapértelmezés szerint a törlés az aktuális sortól kezdődik.

```
*10D
*L
 1: Az EDLIN hibauzeneteinek listaja:
 2: --------------------------------
 3: Cannot edit .BAK file -rename file
 4: (BAK file-t nem lehet szerkeszteni, nevezze at.)
 5: Disk full - write not completed
 6: (A lemez betelt - a tarolas nem fejezodott be)
 7: Entry error
 8: (Hibas parancs)
 9: *Not found
 10: (Nem talalhato)
*
```

Most is a 9. sor maradt aktuális, de a sor tartalma megváltozott.

Töröljük a harmadik sort:

```
*3D
*L
 1: Az EDLIN hibauzeneteinek listaja:
 2: --------------------------------
 3:*(BAK file-t nem lehet szerkeszteni, nevezze at)
 4: Disk full -write not completed
 5: (A lemez betelt -a tarolas nem fejezodott be
 6: Entry error
 7: (Hibas parancs)
 8: Not found
 9: (Nem talalhato)
*
```

Ismét a harmadik sort töröljük, de kihasználjuk, hogy ez az aktuális sor:

```
*D
*L
 1: Az EDLIN hibauzeneteinek listaja:
 2: --------------------------------
 3:*Disk full - write not completed
 4: (A lemez betelt - a tarolas nem fejezodott be)
 5: Entry error
 6: (Hibas parancs)
 7: Not found
 8: (Nem talalhato)
```

```
*
```

Legyünk óvatosak a D parancs alkalmazásakor, mert a törlés visszafordíthatatlan művelet. Biztonságosabb, ha mindig kiírjuk a törlendő sor(ok) számát, és nem hagyatkozunk az alapértelmezésekre.

8.2.6 E (end)- szerkesztés befejezése tárolással

A szöveges file létrehozásának vagy módosításának befejezésekor ki kell lépnünk az EDLIN programból. Ha a kialakított szöveget a lemezen meg akarjuk őrizni, akkor az E (end, vége) parancsot választjuk. Ha a szövegre a jelenlegi formájában nincs szükség, akkor a "Q"-val lépünk ki (lásd a következő pontban).

A "szerkesztés befejezése tárolással" parancs formája:

```
E
```

A parancs hatására a memóriában tárolt szöveg kiíródik a lemezen már korábban megnyitott .$$$ kiterjesztésű file-ba, majd - a sikeres tárolás után - a file neve megkapja az EDLIN indító parancsában előírt, végleges kiterjesztést (lásd a 8.1 pontban). Ha egy meglevő file-t dolgozunk fel, akkor a régi eredeti file is a lemezen marad, de nevének kiterjesztése .BAK-ra változik. Ha ugyanilyen nevű és .BAK végződésű file már volt a lemezen, akkor az most törlődik.

Ha az E parancs kiadásakor a lemezen nincs elegendő hely az új szöveg tárolására, akkor a szövegnek csak egy része kerül ki a lemezre, a memóriában levő szöveg pedig elvész. Ebben az esetben az eredeti file neve nem változik meg (nem lesz .BAK), a részlegesen letárolt új file pedig megtartja az ideiglenes .$$$ kiterjesztést.

Az EDLIN a file-ok végére kiteszi a file-vége jelet (< Ctrl > -Z).

Az E parancs a szöveg tárolása után visszatér a DOS parancs szintjére, amint az a példában is látszik:

```
*E
A>
```

8.2.7 Q (quit)- kilépés a szerkesztésből

Előfordulhat, hogy a memóriában elhelyezkedő szöveg végleges tárolására nincs szükségünk. Ennek néhány lehetséges oka:

- A lemezről beolvasott szöveget csak megnéztük, de nem módosítottuk. Ilyenkor felesleges a tárolás.
- Tévedésből másik file-t olvastunk be, nem azt, amelyiket módosítani akartuk.
- Hibás módosításokat, felesleges törléseket hajtottunk végre, ezért újra szeretnénk kezdeni az eredeti file feldolgozását.

A tárolás nélküli kilépés parancsának formája:

```
Q
```

A Q (quit, kilépés) parancs hatására az EDLIN törli a lemezről az új file számára megnyitott, .$$$ kiterjesztésű file-t. Ha a futást egy meglevő file beolvasásával kezdtük, akkor az eredeti file -a nevének megváltoztatása nélkül- a lemezen marad. A szövegen végrehajtott változtatások elvesznek, archív (.BAK) file nem keletkezik.

Mielőtt az EDLIN visszatérne a DOS-hoz, a biztonság kedvéért felteszi az "Abort edit (Y/N)?" (megszakítja a szerkesztést?) kérdést. Ha valóban tárolás nélkül akarjuk befejezni a futtatást, nyomjuk le az Y billentyűt (Y = yes, igen). Ha bármely más karakterrel válaszolunk, akkor az EDLIN nem tér vissza az operációs rendszerhez és folytathatjuk a szerkesztést, vagy kiadhatjuk az E parancsot.

Az alábbi példában Y-nal megerősítjük a kilépési szándékot:

```
*Q
Abort edit (Y/N)? Y
A>
```

8.2.8 M (move) - sorok áthelyezése

Ha a szövegben egy vagy több sort más helyre akarunk áthelyezni, eddigi ismereteink alapján a következőket tehetnénk: az I parancs kiadása után megfelelő helyre újból beírjuk az áthelyezni kívánt szöveget, majd a régi helyén feleslegessé vált sorokat a D paranccsal kitöröljük. Most megismerkedünk az M (move, áthelyezés) paranccsal, amely lényegesen leegyszerűsíti ezt a feladatot. A parancs általános formája:

[sorszám1],[sorszám2],sorszám3M

ahol

sorszám1 az áthelyezendő szöveg első sorát jelenti,
sorszám2 az áthelyezendő szöveg utolsó sorát jelenti
sorszám3 pedig azt jelenti, amelyik elé a kijelölt szöveget be kell illeszteni.

A harmadik sorszámot meg kell adni, de az első kettő elhagyható. Ha bármelyiket kihagyjuk, az aktuális sor száma lép a ki nem írt sorszám(ok) helyébe.

A szövegrészt önmagán belül nem lehet áthelyezni, ezért a sorszám3 nem eshet a sorszám1 és a sorszám2 közé, továbbá nem lehet egyenlő a sorszám2-vel sem, különben az "Entry error" (hibás parancs) üzenetet kapjuk.

A parancs végrehajtása után az EDLIN újraszámozza a sorokat, és az első áthelyezett sor lesz az aktuális sor.

A példánkban az alábbi szöveggel fogunk dolgozni:

```
1: ; Mintaprogram
2: RECEIVR PROC    FAR
3:         ASSUME CS:SUBSEG
4:         PUBLIC RECEIVR
5:         INT    10H
6:         POP    BX
```

```
  7:        PUSH   BX
  8:        MOV    DX,0
  9:*       MOV    BH,0
 10:        MOV    AH,2
 11: RECEIVR ENDP
```

Tegyük föl, hogy az ötödik és hatodik sor rossz helyen van és át kell őket tenni a tizenegyedik sor elé:

```
*5,6,11M
*L
      1: ; Mintaprogram
      2: RECEIVR PROC   FAR
      3:        ASSUME CS:SUBSEG
      4:        PUBLIC RECEIVR
      5:        PUS    BX
      6:        MOV    DX,0
      7:        MOV    BH,0
      8:        MOV    AH,2
      9:*       INT    10H
     10:        POP    BX
     11: RECEIVR  ENDP
*
```

Hogyan lehetne az első sort áthelyezni az utolsó sor után? Egy lehetséges megoldás:

```
*1,1,#M
*L
      1: RECEIVR PROC   FAR
      2:        ASSUME CS:SUBSEG
      3:        PUBLIC RECEIVR
      4:        PUSH   BX
      5:        MOV    DX,0
      6:        MOV    BH,0
      7:        MOV    AH,2
      8:        INT    10H
      9:        POP    BX
     10: RECEIVR ENDP
     11:*; Mintaprogram
*
```

A # (hashmark, kettőskereszt) az utolsó utáni sorra hivatkozás jele.

Most a tizenegyedik sor az aktuális sor.

Ha az M parancsban elhagyjuk az első két sorszámot, az áthelyezés az aktuális sorra fog vonatkozni:

```
*,,2M
```

```
*L
        1: RECEIVR PROC    FAR
        2:*;Mintaprogram
        3:         ASSUME  CS:SUBSEG
        4:         PUBLIC  RECEIVR
        5:         PUSH    BX
        6:         MOV     DX,0
        7:         MOV     BH,0
        8:         MOV     AH,2
        9:         INT     10H
       10:         POP     BX
       11: RECEIVR ENDP
*
```

8.2.9 C (copy)-sorok átmásolása

A C (copy, másolás) parancs hasonlít az előző pontban bemutatott M (áthelyezés) parancshoz, de a legfőbb különbség, hogy a C parancsban kijelölt sorok a helyükön maradnak, nem törlődnek a másolás után. Ezzel lehet tehát egy vagy több sort a szövegben megismételni. A parancs általános formája:

[sorszám1],[sorszám2],sorszám3[,számláló]C

ahol
sorszám1 az átmásolandó szöveg első sorát jelenti,
sorszám2 az átmásolandó szöveg utolsó sorát jelenti,
sorszám3 azt a sort jelzi, amelyik elé a kijelölt szöveget be kell másolni,
számláló pedig azt mutatja, hányszor kell a szöveget a másoláskor megismételni.

A paraméterek közül csak a harmadik sorszámot kötelező megadni. A ki nem írt paraméterekre a következő alapértelmezés érvényesül: a sorszám1 és a sorszám2 helyett az EDLIN az aktuális sor számát feltételezi. A számláló alapértelmezése: 1, tehát a szöveg egy példányban másolódik a kijelölt helyre.

A szövegrészt önmagán belül nem lehet átmásolni, ezért a sorszám3 nem eshet a sorszám1 és a sorszám2 közé, továbbá nem lehet egyenlő a sorszám2-vel sem, különben az "Entry error" (hibás parancs) üzenetet kapjuk.

A parancs végrehajtása után az EDLIN újra számozza a sorokat, és az első átmásolt sor lesz az aktuális sor.

A példánkban az alábbi szöveget fogjuk felhasználni:

```
1: Greer harmadik torvenye:
2: ************************
3: A szamitogepprogram
4:*nem kivansagaid, hanem
5: utasitasaid szerint
6: mukodik.
```

Duplázzuk meg a szöveg harmadiktól hatodikig terjedő sorait. A megismételt szövegrész közvetlenül a hatodik sor után kezdődjön:

```
*3,6,#C
*L
 1: Greer harmadik torvenye:
 2: *************************
 3: A szamitogepprogram
 4: nem a kivansagaid, hanem
 5: utasitasaid szerint
 6: mukodik.
 7:*A szamitogepprogram
 8: nem kivansagaid, hanem
 9: utasitasaid szerint
 10: mukodik.
*
```

Ismételjük meg az aktuális (hetedik) sort két példányban, a tizedik sor előtt:

```
*,,10,2C
*L
 1: Greer harmadik torvenye:
 2: ************************
 3: A szamitogepprogram
 4: nem kivansagaid, hanem
 5: utasitasaid szerint
 6: mukodik.
 7: A szamitogepprogram
 8: nem kivansagaid, hanem
 9: utasitasaid szerint
 10:*A szamitogepprogram
 11: A szamitogepprogram
 12: mukodik.
*
```

A példában kihasználtuk a sorszám1-re és a sorszám2-re vonatkozó alapértelmezéseket. Kiírtuk viszont a számláló paramétert, melynek hatására a kijelölt sor két példányban másolódott át.

8.2.10 S (search) - karaktersorozat keresése

Az S (search, keresés) parancs kiadásakor az EDLIN megkeresi a szövegben a megadott karaktersorozatot, és megjeleníti azokat a sorokat, amelyekben a keresett szövegrész előfordul. E parancs igen hasznosnak bizonyul, ha például egy programban az ALAPBER változó valamennyi előfordulását ellenőrizni akarjuk, vagy ha az Egri csillagok című regényben a Dobó Istvánra vonatkozó mondatokat szeretnénk kielemezni.

A parancs általános formája:

```
[sorszám1][,sorszám2][?]S[karaktersorozat]
```

ahol

sorszám1 jelöli azt a sort, amelynél a keresés kezdődik,
sorszám2 jelöli azt a sort, amelynél a keresés befejeződik,
? jelzi, hogy nem kell befejezni a keresést az első találat után,
karaktersorozat helyébe pedig a keresett szövegrészt kell írni.

A megadott karaktersorozat előfordulásait az EDLIN csak a sorszám1 és a sorszám2 által kijelölt tartományban keresi meg. A két sorszámot nem kötelező kiírni.

Ha a sorszám1-et elhagyjuk, akkor a keresés az aktuális sor utáni sortól kezdődik. Ha a sorszám2-t hagyjuk el, akkor a keresés a memóriában elhelyezkedő szöveg végéig fog tartani. Ha egyik sorszámot sem írjuk ki, akkor mindkét sorszámra az előbb leírt alapértelmezés érvényesül.

Ha a kérdőjelet nem adjuk meg, akkor az EDLIN megjeleníti az első olyan sort, amelyben szerepel a kijelölt karaktersorozat és ezzel be is fejezi a keresést. (Ha ilyen sort nem talált, akkor a "Not found" (nem található) üzenetet írja ki. Ha a parancsban szerepel a kérdőjel-paraméter, akkor az EDLIN az első találatot tartalmazó sor után felteszi az "O.K.?" kérdést és a keresést válaszunktól függően folytatja. Ha az Y (yes, igen) vagy az < ENTER > billentyűt nyomjuk le, akkor a keresés befejeződik. Bármely más válasz esetén az EDLIN tovább keres, és a következő olyan sort jeleníti meg, amelyben szintén szerepel a kijelölt szövegrész.

A kereséseket mindaddig folytathatjuk az "O.K.?" kérdésre adott nemleges válasszal, amíg a "Not found" üzenetet nem kapjuk.

A sikeres keresés után a megadott karaktersorozatot tartalmazó sor lesz az aktuális sor. Ha az EDLIN ilyet nem talál, akkor az aktuális sor nem változik.

A karaktersorozat beírásakor pontosan kell idéznünk a keresett szövegrészt; a nagybetűs ill. kisbetűs írásmód ebből a szempontból különbséget jelent. (Ha az "EDLIN" szót keressük, a gép nem jelez találatot az "edlin" szónál.) A karaktersorozatot közvetlenül az S betű után kell írni, mert az esetleges közbenső szóközöket az EDLIN a karaktersorozat részének tekinti. A karaktersorozat végét általában az < ENTER > lenyomásával jelezzük (kivéve, ha az S paranccsal egy sorban további utasításokat akarunk írni, ilyenkor az < F6 > vagy a < Ctrl-Z > billentyűket használjuk az < ENTER > helyett).

Ha a parancsban nem adjuk meg a keresendő karaktersorozatot, akkor az EDLIN a legutóbbi S, ill. R (replace, helyettesítés) parancsban szereplő szövegrész előfordulásait fogja keresni. Ez az ismétlési eljárás nagyon hasznos, mert sok írásmunkát takaríthat meg. Az S parancs hatására kijelzett sorokat az EDLIN aktuálissá teszi, így ezeket a "."(pont) utasítással lehet átszerkeszteni.

A keresési parancs illusztrálására a következő szöveget fogjuk felhasználni:

```
1: Cahm axiomaja
2: *******************
```

```
3: (az EDLIN-re is ervenyes)
4:*Ha mar minden kiserlet
5: csodot mond, olvasd el
6: a hasznalati utasitast
7:
8: Sattinger torvenye
9: ********************
10: (a szamitogepre is vonatkozik)
11: Jobban mukodik, ha
12: bedugod a konnektorba.
```

Keressük meg az "a" névelő előfordulásait. A keresett karaktersorozat ne az "a" betű legyen, mert ez a szavak belsejében található "a" betűket is kiválasztaná. Ezért az "a" után egy szóközt hagyunk ki, így csak azokat az "a"-kat kapjuk meg, amelyek a szavak végén állnak.

```
*1?Sa <Enter>
 4:*Ha mar minden kiserlet
O.K.? N
 6: a hasznalati utasatastl
O.K.? N
 10: (a szamitogepekre is vonatkozik)
O.K.? N
 12: bedugod a konektorba.
O.K.? N
Not found
*
```

A szövegben előforduló valamennyi névelőre szükségünk volt, azért előírtuk, hogy a keresés az első sornál kezdődjön. A második sorszámot elhagytuk, így a keresés a szöveg végéig tartott. A kérdőjelet azért adtuk meg az S parancsban, hogy a keresés ne álljon le az első találat után.

A negyedik sorban a "Ha" szót találta meg a gép, az első és a tizenegyedik sorban viszont azért nem jelzett találatot a szóvégi "a" betűknél, mert azok után nem található szóközjel, csak a sor végét jelző kocsivissza-karakter.

Az EDLIN "O.K.?" kérdésére mindig nemleges választ adtunk, hogy folytatódjon a keresés. Amikor az EDLIN a szöveg végére ért és további egyezéseket nem talált, kiírta a "Not found" üzenetet. A fenti műveletek elvégzése után a negyedik maradt az aktuális sor, mert az utolsó keresés sikertelen volt, a "Not found" üzenettel végződött.

A következő példában keressük meg a második cím aláhúzását (a csillagokból álló sort), és helyettesítsük kötőjelekkel. A parancs kiadásakor emlékszünk arra, hogy ez a sor a negyedik és a tizedik sor között helyezkedik el:

```
*4,10S***
 9:******************
*
```

```
  9:******************
  9:------------------
*L
  1: Cahn axiomaja
  2: *****************
  3: (az EDLIN-re is ervenyes)
  4: Ha mar minden kiserlet
  5: csodot mond, olvasd el
  6: a hasznalati utasitast!
  7:
  8: Sattinger torvenye
  9:*-----------------
 10: (szamitogopre is vonatkozik)
 11: Jobban mukodik, ha
 12: bedugod a konnektorba.
*
```

A szerkesztő parancsban a "." (pont) alkalmazásakor kihasználtuk, hogy az utoljára megtalált sor lett az aktuális sor. A sormódosítást úgy végeztük el, hogy a régi tartalom helyett tizennyolc kötőjelet írtunk be. Az átszerkesztés után is a kilencedik sor maradt az aktuális.

8.2.11 R (replace) - karaktersorozat helyettesítése

Az R (replace, helyettesítés) parancs végrehajtásának első fázisa, a keresés, megegyezik az S parancsnál leírtakkal. A második fázisban viszont az R parancs módosítja a megtalált karaktersorozatot; helyettesíti az általunk megadott szöveggel. Egy gyakorlati példa; az R paranccsal egy lépésben módosíthatjuk a programunkban a CIM változó valamennyi előfordulásánál a változó nevét LAKCÍM-re. A parancs általános formája :

[sorszám1][,sorszám2][?]R[karaktersor][< F6 > karaktersor2]

ahol

sorszám1 jelöli azt a sort, amelynél a keresés és helyettesítés kezdődik,
sorszám2 jelöli azt a sort, amelynél a keresés és helyettesítés befejeződik,
? megadásakor a helyettesítések nem automatikusan mennek végbe, hanem az EDLIN minden találatnál megkérdezi, hogy elvégezze-e a helyettesítést,
karaktersor1 helyébe a keresett szövegrészt kell beírni,
karaktersor2 helyébe pedig a keresett szövegrész helyére írandó új szöveget kell írni.

A megadott karaktersorozat előfordulásait az EDLIN csak a sorszám1 és a sorszám2 által kijelölt tartományban keresi meg. A két sorszámot nem kötelező kiírni. Ha a sorszám1-et elhagyjuk, akkor a keresés és helyettesítés az aktuális sor utáni sortól kezdődik. Ha a sorszám2-t hagyjuk ki, akkor a keresés és helyettesítés a memóriában elhelyezkedő szöveg végéig fog tartani. Ha egyik sorszámot sem írjuk ki, akkor mindkét sorszámra az előbb leírt alapértelmezés érvényesül.

Ha a kérdőjelet nem adjuk meg, akkor az EDLIN - a kijelölt tartományban - a karaktersor1 valamennyi előfordulását kicseréli a karaktersor2-ben megadott szövegre. A módosított sorok eközben megjelennek a képernyőn. Ha az EDLIN egy módosítandó szöveget sem talál, akkor a "Not found" (nem található) üzenetet írja ki.

Ha a parancsban szerepel a kérdőjel-paraméter, akkor az EDLIN a karaktersor1 minden helyettesítése után leáll, és felteszi az "O.K.?" kérdést. Ha a helyettesítéssel egyetértünk, akkor nyomjuk le az Y (YES) vagy az < ENTER > billentyűt.Bármely más válasz esetén az EDLIN visszaállítja a karaktersor1-ben megadott eredeti tartalmat. Mindkét esetben a keresés és helyettesítés tovább folytatódik a kijelölt tartomány végéig.

A helyettesítések végrehajtása után az utolsó módosított sort megelőző sor lesz az aktuális sor.

A karaktersor1 és a karaktersor2 beírásakor nagyon pontosan kell idézni a keresett és helyettesítő szövegeket. Ne felejtsük el a kis- és nagybetűs írásmód közötti különbséget. A karaktersor1-et közvetlenül az R betű után, a karaktersor2-t pedig az < F6 > után kell írni, mert az esetleges közbenső szóközöket az EDLIN a karaktersorozatok részének tekinti. A karaktersor1 végét az < F6 > vagy a < Ctrl > -Z billentyűkkel kell jelezni (vagy az < ENTER > -rel, ha a karaktersor2 nem szerepel a parancsban). A karaktersor2 végén általában az < ENTER > -t használjuk (kivéve ha az R paranccsal egy sorban további utasításokat akarunk írni, ilyenkor az < F6 > vagy a < Ctrl > -Z billentyűket nyomjuk le az < ENTER > helyett).

Ha a parancsban az < F6 > billentyű lenyomása után nem írunk be helyettesítő szöveget (a karaktersor2-t), akkor az EDLIN törli a karaktersor1 valamennyi előfordulását a kijelölt tartományban. Ha az < F6 > -ot sem adjuk meg, akkor a gép a legutóbbi R parancsban szereplő helyettesítő szöveget fogja felhasználni. Ha mindkét karaktersorozatot elhagyjuk, akkor a keresési szöveg (karaktersor1) helyébe a legutóbbi S, ill. R parancsban szereplő szövegrész lép, a helyettesítő szöveg (karaktersor2) helyett pedig az EDLIN a legutóbbi R parancs helyettesítő szövegét használja föl (ha volt ilyen).

A helyettesítési parancs alkalmazását a következő szövegen mutatjuk be:

```
1:        IF KERESET > 15300
2:           MOVE 0.15 TO NYUGJAR
3:        MULTIPLY KERESET BY NY.
4:        MOVE NY TO NYUGJAR.
5:        SUBTRACT NYUGJAR FROM KERESET
6:*          GIVING KIFIZ.
```

A példában egyszer hibásan a NYUGJAR változót írtuk NY változó helyett (a második sorban). Automatikus helyettesítést azonban nem végezhetünk, mert a NYUGJAR többi előfordulását nem kell megváltoztatni:

```
*1?RNYUGJAR<F6>NY
 2:          MOVE 0.15 TO NY
*O.K.? Y
 4:        MOVE NY TO NY.
*O.K.? N
```

```
  5:         SUBTRACT  NY FROM KERESET
*O.K.? N
*L
  1:*         IF KERESET > 153000
  2:             MOVE 0.15 TO NY.
  3:          MULTIPLY KERESET BY NY.
  4:          MOVE NY TO NYUGJAR.
  5:          SUBTRACT NYUGJAR FROM KERESET
  6:             GIVING KIFIZ.
*
```

Az R parancsban elhagytuk a második sorszámot, ezért a végrehajtás az utolsó sornál fejeződött be. Az "O.K.?" kérdésekre adott nemleges válaszok esetén az EDLIN visszaállította a sorok eredeti tartalmát. Módosítsuk végül a teljes szövegben a KERESET változó nevét JOVEDELEM-re. Az automatikus helyettesítés módszerét alkalmazhatjuk;

```
*1RKERESET<F6>JOVEDELEM
  1:*          IF JOVEDELEM > 15300
  3:           MULTIPLY  JOVEDELEM BY NY.
  5:           SUBTRACT NYUGJAR FROM JOVEDELEM
*L
  1:           IF JOVEDELEM > 15300
  2:              MOVE 0.15 TO NY.
  3:           MULTIPLY JOVEDELEM BY NY.
  4:*          MOVE NY TO NYUGJAR.
  5:           SUBTRACT NYUGJAR FROM JOVEDELEM
  6:              GIVING KIFIZ.
*
```

8.2.12 T (transfer) - sorok beszúrása lemezről

A T (transfer, beszúrás, beolvasás lemezről) parancs segítségével egy lemezes adatállomány tartalmát lehet a memóriában elhelyezkedő sorok közé beilleszteni. Ilyen módon több file szövegét tudjuk egybeszerkeszteni. A parancs általános formája:

```
[sorszám]T[d:]file-név[.kit]
```

ahol

sorszám jelöli azt a sort, amelyik elé a file szövegét be kell illeszteni,
[d:]file-név[.kit] pedig a beolvasandó file specifikációja.

A sorszámot nem kötelező kiírni. Ha elhagyjuk, akkor a beillesztés az aktuális sor elé történik. A beolvasandó file specifikációjában nem lehet utat (path) megadni; a file-t az EDLIN mindig az adott egység aktuális könyvtárában keresi.

A művelet végrehajtása után az EDLIN újraszámozza a sorokat. Az aktuális sor az első beolvasott sor lesz.

Ha a file beolvasása közben megtelik a memória, akkor a "Not enough room to merge the entire file" (kevés a memória a teljes file beolvasásához) hibaüzenetet kapjuk.

A példánkban a következő szöveggel indulunk:

```
 1:         READ (*,6001) IE,IV
 2: 6001    FORMAT (2I3)
 3:         CALL GYOK (IE,IV)
 4:*        END
```

Tegyük fel, hogy az aktuális meghajtón az SQRT nevű állományban helyezkedik el az a szubrutin, amelyet a fenti program végéhez szeretnénk csatolni. Ebhez a # jelet használjuk fel, amely az utolsó utáni sor szimbóluma:

```
*#TSQRT
*L
  1:       READ (*,6001)IE,IV
  2: 6001  FORMAT (2I3)
  3:       CALL GYOK (IE,IV)
  4:       END
  5:*      SUBROUTINE GYOK (K,L)
  6:       WRITE (*,6000) (I,I=0,9),
  7:      X  (I,(SQRT(I+0.1*J),J=0,9),I=K,L)
  8; 6000  FORMAT (3H  ,10(6H  0.,I1
  9:      X  /,10(/,2HO ,I2,1X,10(1X,F6.3)))
  10:        END
*
```

A beolvasott sorokat az EDLIN a memóriában elhelyezkedő sorok után illesztette.

8.2.13 W (write) - sorok kiírása lemezre

A W (write, kiírás lemezre) parancsot akkor használjuk, ha a feldolgozandó file nem fér be a számítógép memóriájába és a szöveg létrehozását, módosítását több részletben kell elvégeznünk. Egyébként a "W" használata nélkül, az E paranccsal írhatjuk ki a teljes file-t a lemezre.

Ha az EDLIN indító parancsában megnevezett file már eleve túl hosszú volt, akkor az EDLIN csak annyi sort olvas be, amennyi a szabad memória 75%-át tölti ki. A memóriában elhelyezkedő szöveget (vagy annak egy részét) ki kell íratni a lemezre, ha

- a memória teljesen megtelt, és/vagy
- a szöveg további, még a lemezen elhelyezkedő részeit akarjuk a memóriába beolvasni és feldolgozni.

A parancs általános formája:

```
[n]W
```

ahol az n helyébe egy pozitív egész számot írhatunk.

A W parancs hatására az EDLIN kiírja a lemezre (a .$$$ kiterjesztésű állományba) a memóriában elhelyezkedő szöveg első n sorát. Ha nem adtuk meg az n paramétert, akkor az EDLIN addig írja a sorokat a lemezre, ameddig a memória telítettsége 25%-ra nem csökken. (Ha a szöveg a memóriának kevesebb mint a 25%-át foglalta el, akkor a paraméter nélküli W parancs hatástalan marad.)

A lemezre kiírt sorok törlődnek a memóriából, további feldolgozásuk csak az EDLIN újraindítása után lehetséges. A memóriában maradt sorok egytől kezdve újraszámozódnak. Az aktuális sor az új első sor lesz.

A memória felszabadítása után beolvashatjuk (és feldolgozhatjuk) az állomány következő részét. Erre a célra az A parancs szolgál, melyet a következő pontban ismertetünk.

Ha kíváncsiak vagyunk, hány sor található a memóriában, adjuk ki a

```
#I
```

parancsot. Ekkor megjelenik az utolsó sor számánál eggyel nagyobb sorszám. Nem akarunk újabb sorokat beírni, ezért a beszúró módból mindjárt ki is lépünk a <Ctrl>-<Break>-kel.

Tegyük fel, hogy memóriában 527 sor helyezkedik el, de a szöveg vége még a lemezen található. A további sorok beolvasása előtt helyet kell csinálnunk a tárban.

Néhány példa erre a műveletre:

```
*W
```

A paraméter nélkül kiadott parancs felszabadította a memória 75%-át

```
*600W
```

A létező sorok számánál nagyobb értéket adtunk meg. Az EDLIN az 527 sort kiírta a lemezre, a memória jelenleg üres.

```
*300W
```

Az 1-300. sorok kerültek ki a lemezre, az eddigi 301. sor lett az új első (és aktuális) sor. A tárban hely szabadult fel a további sorok beolvasására.

8.2.14 A (append) - sorok hozzáfűzése lemezről

Az A (append, sorok hozzáfűzése) parancsot az előző pontban leírt W paranccsal együtt használjuk, ha a módosítandó szöveg túl hosszú ,és feldolgozását csak részletenként lehet elvégezni. Az A parancs - feltéve, hogy elegendő hely van a gép memóriájában - a lemezes adatállomány következő részét beolvassa és hozzáilleszti a tárban elhelyezkedő szöveg végéhez. A parancs általános formája:

```
[n]A
```

ahol n helyébe egy pozitív egész számot írhatunk.

Ha megadtuk az n paramétert, akkor az EDLIN az indító parancsában megnevezett lemezes adatállományból beolvassa a következő n darab sort. Ha a paramétert nem írtuk ki, akkor az EDLIN addig olvassa be a sorokat, ameddig a memória eléri a 75%-os telítettséget. Amennyiben a telítettség 75%-on felül volt, nem történik beolvasás.

Az újonnan beolvasott sorok mindig a tárban elhelyezkedő sorok után íródnak (sorszámaikat is ennek megfelelően kapják). Az aktuális sorszám nem változik.

Az EDLIN az "End of input file" (input file vége) üzenettel jelzi, amikor a feldolgozandó szöveg utolsó sorait is beolvasta a memóriába.

Az A parancs végrehajtása után elvégezhetjük a szöveg következő részének módosítását, átszerkesztését. A módosítás után a memóriában található sorokat a W paranccsal vihetjük ki a lemezre, hogy helyünk legyen a szöveg következő sorainak beolvasására.

Nézzük meg az alábbi példát:

```
A>EDLIN HOSSZU.TXT
*
```

Nem jelent meg az "End of input file" üzenet, tehát a megnevezett szöveg nem fér el egyszerre a memóriában. Vigyük ki lemezre az első 200 sorát, és olvassuk be a következő 200-at:

```
*200W
*200A
*
```

Most a memóriában az eredeti szöveg 201-től kezdődő sorai helyezkednek el (természetesen 1-től számozva). A szöveg ellenőrzése, módosítása után olvassuk be az újabb sorokat:

```
*W
*A
End of input file
*
```

A W parancs annyi sort írt ki a lemezre, amennyi a memória telítettségét 25%-ra csökkentette. A paraméter nélküli A parancs a tárat 75%-ra töltötte volna fel, de közben véget ért az input szöveg. Most tehát az eredeti szöveg utolsó sorai találhatók a memóriában.

9. A nyomkövető-hibakereső program (DEBUG)

Az alábbiakban bemutatott DEBUG rendszerprogram elsősorban a gyakorlott programozók számára készült, akik jól ismerik a számítógép felépítését és mikroprocesszorának működésmódját. A kevesebb tapasztalattal rendelkező felhasználóknak is javasoljuk, hogy tekintsék át, milyen feladatokra alkalmas a DEBUG. Az egyszerűbb parancsokat ők is bátran kiadhatják.

A DEBUG nyomkövető-hibakereső program lehetővé teszi, hogy a számítógép regisztereiben, operatív tárában (memóriájában) és lemezes háttértárolóiban található (gépi kódú) programok tesztelésére és javítására használjuk. A DEBUG futtatásakor a következő alapvető funkciók állnak rendelkezésünkre:

- a mikroprocesszor regisztereinek kiíratása és módosítása,
- az operatív tár tartalmának megjelenítése és módosítása,
- assembly nyelvű utasítások írása és az operatív tárba való elhelyezése,
- a memóriában található program visszafordítása assembly nyelvre (disassemblálás),
- az operatív tárban elhelyezkedő gépi kódú program futtatása (esetleg a kijelölt pontokon való megállással vagy lépésenkénti végrehajtással),
- adott tartalom keresése a memóriában,
- a memória kijelölt területének feltöltése a megadott tartalommal,
- a memória két területének összehasonlítása,
- az operatív tár kijelölt területének átmásolása,
- lemezen elhelyezkedő file-ok, ill. kijelölt lemezes szektorok betöltése a memóriába,
- a memória adott területének kiírása egy lemezes file-ba vagy kijelölt lemezes szektorokba,
- beolvasás a kijelölt portról (input-output csatornáról) és az adat megjelenítése,
- adott byte kivitele a kijelölt portra,
- hexadecimális összeadás és kivonás elvégzése.

A felsorolt műveletek segítségével a gépi kódú (assembly nyelven írt, illetve magasszintű nyelvről lefordított) programok kipróbálása a következő lépésekben történhet. Először betöltjük a programot a lemezről a memóriába. Ezután a kritikus pontokra helyezett megszakító utasításokkal (vagy lépésenkénti végrehajtással) lefuttatjuk a programot, és a megszakítások elérésekor megvizsgáljuk azt az információt, amelyet a DEBUG kiír a programvégrehajtás állásáról (regiszterek, fontosabb memóriaterületek tartalmát stb.). Szükség esetén a DEBUG mini assemblerével javításokat is végezhetünk a programon.

A számítógép 8088-as (vagy azzal kompatibilis) mikroprocesszorának utasításait nem ismerő programozók is végrehajthatnak kisebb módosításokat a programokon. Ha megtaláljuk az idegen nyelven kommunikáló programokban a futáskor megjelenített szövegeket, könnyen átírhatjuk azokat magyarra. A másik hasznos módosítás: ha egy program csak az "A" betűjelű floppy egységről hajlandó egy file-t beolvasni, mi pedig a "C" jelű fix lemezt akarjuk használni, keressük meg a programban a file-ra való hivatkozást (például: A:HELP.TXT), majd írjuk át az A betűt C-re.

A teljesség kedvéért megjegyezzük, hogy a DEBUG-nál nagyobb tudású és kényelmesebb kezelésű nyomkövető-hibakereső programok is forgalomban vannak. Utóbbiak néhány többletszolgáltatása: teljes képernyős szerkesztés, memóriacímekre szimbólikus nevekkel való hivatkozási lehetőség stb.

A DEBUG működésének teljes megértéséhez szükséges a mikroprocesszor regisztereinek és utasításkészletének, valamint a számítógép felépítésének ismerete. Az irodalomjegyzékben felsorolunk olyan könyveket, amelyek ezekkel a kérdésekkel részletesen foglalkoznak.

9.1 A DEBUG indítása

A DEBUG program a DOS rendszerlemezen található, DEBUG.COM néven, de szükség esetén bármely más lemezre átmásolhatjuk. Ha a program az aktuális lemezegység aktuális alkönyvtárában (vagy a PATH parancsban kijelölt úton) elérhető, akkor a következő paranccsal indíthatjuk el:

DEBUG [[*d:*][*út*]*file-név*[*.kit*] [*par1*] [*par2*]]

ahol

[*d:*][*út*]*file-név*[*.kit*]

a megvizsgálni kívánt adat- vagy program-file specifikációja,

par1 és *par2*

pedig a megnevezett program futtatásához szükséges paraméterek.

A jelölésekből látszik, hogy a file-specifikáció és a paraméterek kiírása sem kötelező. Ha a file-specifikációt megadjuk az indító parancsban, akkor a DEBUG első dolga lesz, hogy betölti a memóriába a megnevezett file-t. Ha a tesztelni kívánt program paramétereket vár a parancs-sorban, akkor azokat a szokott módon a program neve után kell írni. A BASIC interpretert például a

```
BASIC
```

paranccsal indíthatjuk. A

```
BASIC RAKETA
```

parancs hatására pedig a BASIC interpreter az indulása után mindjárt betölti a RAKETA.BAS programot, és el is kezdi futtatni. Ha viszont nemcsak futtatni akarjuk a programot, hanem a DEBUG-gal nyomon is szeretnénk követni, hogyan futtatja a BASIC a RAKETA programot, akkor a következő indító parancsot kell kiadnunk:

```
DEBUG BASIC.COM RAKETA
```

Az alábbi példában az A jelű floppy SZOVEG nevű alkönyvtárából töltjük be a LEVEL.DOC nevű file-t (a DEBUG program a C lemezen helyezkedik el):

```
C:DEBUG A:\SZOVEG\LEVEL.DOC
```

Ha a file-specifikációt nem adjuk meg, akkor a DEBUG nem tölt be semmit a memóriába, így csak a tárba korábban bekerült adatokat vizsgálhatjuk, módosíthatjuk. Az N (name) és az L (load) parancsok segítségével azonban még lehetőségünk lesz file-ok betöltésére.

A DEBUG indulásakor a regiszterek és a flag-ek (jelzők) a következő értékeket kapják:

- A négy szegmensregiszter (CS, DS, ES és SS) a szabad memóriaterület aljára mutatnak, azaz a DEBUG program fölötti első szegmensre.
- Az utasításmutató (IP) értéke hexadecimális 0100.
- A veremmutató (SP) vagy a szegmens végére mutat, vagy a COMMAND.COM tranziens részének aljára (amelyik cím alacsonyabb). A 6-os relatív címen (offset-en) található szegmensméret-mutatót a DEBUG hexadecimális 100-zal csökkenti, mert ekkora területet kap a verem.
- A további regiszterek (AX, BX, CX, DX, BP, SI, DI) a nulla értéket kapják, kivéve, ha az indító parancsban egy file betöltését is előírtuk. Utóbbi esetben a BX és a CX regiszterek a betöltött file hosszát adják meg, byte-okban kifejezve (a hossz alacsony helyiértékű része a CX, magasabb helyiértékű része a BX regiszterbe kerül).
- A flag-ek (jelzőbitek) törlődnek, értékük nulla lesz, kivéve a megszakítás- (interrupt) flag-et, amely 1-re állítódik (megszakítás engedélyezve).
- A lemez-puffer (Disk Transfer Area) kezdőcíme a kódszegmensben hexadecimális 80 lesz.

További megjegyzések a betöltött file-lal kapcsolatban:

A betöltött program futása során nem igényelhet további memóriát, mivel a DEBUG már valamennyi szabad területet lefoglalta a program számára.

Ha a betöltött program a .EXE kiterjesztéssel rendelkezik, akkor a DEBUG elvégzi a futtatáshoz szükséges előkészítő tevékenységeket. Végrehajtja az előírt relokációt (a program áthelyezhetőségéből adódó címmódosításokat) és beállítja a file-ban megadott értékekre a szegmensregisztereket, a veremmutatót és az utasításmutatót. A DS és az ES regiszterek azonban a legalacsonyabb című szegmensben elhelyezkedő ún. programszegmens-prefixre (PSP) mutatnak, a BX és a CX regiszterek pedig a program méretét tartalmazzák (ez kisebb, mint a file mérete).

Ha a programszerkesztőnek (LINK) megadtuk a /H paramétert, akkor a DEBUG a programot a memória lehető legmagasabb című részére helyezi el.

Ha a betöltött file .HEX kiterjesztést visel, akkor a file-ban ASCII-kódolású hexadecimális számjegyeknek kell szerepelniük (egy byte-on egy hexadecimális számjegy, karakteres ábrázolással). A DEBUG a hexadecimális jegyeket a betöltés során binárissá alakítja át.

9.2 A DEBUG parancsai

A DEBUG elindítása után egy kötőjel (-) jelzi a képernyőn, hogy a program a parancsainkat várja. A kötőjelet a DEBUG felszólító jelének, prompt-jának szokták nevezni. A kötőjel mellé kell beírnunk a DEBUG speciális parancsait.

A DEBUG parancsai egy betűből állnak, de a parancsokat általában további paraméterek követik. A parancsok beírásakor a kis- és a nagybetűket is használhatjuk. A parancsok végrehajtása csak akkor kezdődik el, amikor az < ENTER > billentyűt lenyomjuk.

A parancsokat és a paramétereket határoló jelekkel lehet egymástól elválasztani. Az elválasztás azonban csak két egymást követő hexadecimális szám között kötelező. Határoló jelnek szóközt vagy vesszőt használjunk.

A <Ctrl-Break> billentyűk együttes lenyomása félbeszakítja az éppen végrehajtás alatt álló parancsot.

A DEBUG futtatásakor szabadon használhatjuk a DOS szerkesztő billentyűit (ezeket a 2.4 pontban ismertettük).

Ha hibásan írtunk be egy DEBUG parancsot, akkor a program az "error" (hiba) szöveget jeleníti meg, és az első, értelmetlennek talált karakterre mutat. Ilyenkor a parancsot - hibátlanul - újra be kell írnunk.

A legtöbb parancsban paraméterként tárcímeket kell megadnunk. A címek írásmódját az alábbi általános forma mutatja:

[*szegmens*:]*relcím*

ahol

szegmens egy (tizenhat byte-os határon kezdődő) társzegmens kezdőcímét adja meg,
relcím pedig a szegmens kezdőcímétől számított relatív cím (offset).

Az ilyen módon felírt címből a tényleges (abszolút) címet úgy számíthatjuk ki, hogy a szegmenscímet megszorozzuk tizenhattal, majd hozzáadjuk a relatív címet.

A szegmens helyére nemcsak egy konkrét címet, hanem egy szegmensregiszter nevét (CS, DS, ES, SS) is beírhatjuk. Ebben az esetben a szegmens kezdőcímét a kijelölt regiszter tartalma határozza meg. Ha a szegmenscímet nem adjuk meg (ilyenkor a kettőspontra sincs szükség), akkor az egyes parancsoknál leírt alapértelmezés érvényesül.

A szegmens- és a relatív címet maximum négyjegyű hexadecimális számok formájában kell beírni. Vigyázzunk, hogy csak fizikailag létező tárcímekre hivatkozzunk, különben értelmetlen eredményt kapunk.

Néhány példa címek megadására:

1A3

Ez a parancstól függő alapértelmezés szerinti szegmensen belül az 1A3 relatív címet jelenti. (Az alapértelmezéseket az egyes parancsok leírásánál ismertetni fogjuk.)

12C8:A236

```
Az abszolút cím kiszámítása:   12C80
                             + A236
                               -----
                               1CEB6
```

CS:56

Ez a CS regiszter tartalma által meghatározott szegmensen belüli 56-os relatív címre utal.

Egyes parancsokban nem egyetlen tárcímet kell megadnunk, hanem a memóriacímek egy tartományát, intervallumát. Az intervallum egyik lehetséges felírása (általános alakban):

```
cím cím
```

ahol az első címet a fent ismertetett valamelyik formában kell megadni, a második cím pedig csak egy relatív címből állhat. A két címet szóköz vagy vessző választja el. Az intervallum ennél a felírásnál az elsőtől a második címig terjed (beleértve a határokat is). Példák memória-intervallumok megadására:

```
CS:1000 1F00
2000:1234,1235
F000 FFFF
```

Az intervallumok megadásának második formájában a kezdőcím mellett az intervallum hossza szerepel (byte-ban kifejezve):

```
cím L hossz
```

ahol a hosszat hexadecimálisan kell megadni. A hossz értékének maximuma hexadecimális 10000 lehet, de a négyjegyű ábrázolási korlát miatt az 10000 érték helyett 0-t kell írni. Néhány példa:

```
ES:100 L 2F
A000 L 0
```

Számos parancs szintaktikus leírásánál találkozunk majd az adatlista fogalmával. Az adatlista szóközzel vagy vesszővel elválasztott byte- és string-értékekből áll. A byte-okat egy- vagy kétjegyű hexadecimális számmal ábrázolhatjuk. A stringértékek idézőjelek vagy aposztrófok közé zárt karakterek lehetnek. Példák az adatlistára:

```
5C "xyz" 4 11 'abcd'
'ez egy "string"',FF,0
```

A stringeknél az egyes byte-ok értékét a beírt karakterek ASCII-kódjai határozzák meg. Mint láttuk, az aposztrófok között megengedett az idézőjelek használata, és ez fordítva is igaz. Ha azonban a stringben a határoló jellel megegyező karaktert akarunk szerepeltetni, akkor az adott jelet meg kell duplázni:

```
'John''s cat'
```

9.2.1 D (dump) - memóriatartalom kiíratása

A D (dump, tárkiíratás) parancs segítségével tekinthetjük meg a képernyőn a memória kijelölt részét. A parancs általános formája:

D [*cím*]

vagy

D [*intervallum*]

Ha a parancsot paraméter nélkül adjuk ki, akkor a DEBUG az előző D parancs által kiírt utolsó byte utáni címet tekinti kezdőcímnek. Ha csak a relatív cím (offset) részét írtuk ki, akkor a DS regiszter tartalma adja a szegmenscímet. Ha nem volt megelőző D parancs, akkor DS:100-tól kezdődik a kiírás.

Ha a parancsban egy intervallumot specifikáltunk, akkor az intervallumhoz tartozó byte-okat jeleníti meg a DEBUG. Ha csak a kezdőcímet adtuk meg (explicit vagy implicit módon), akkor hexadecimálisan 80 (decimálisan 128) byte íródik ki a képernyőre.

A képernyő baloldalán (a cím mellett) az egyes byte-ok hexadecimális formában, a jobboldalon pedig ASCII-karakterek formájában jelennek meg. A nyomtatbatatlan karakterek helyett egy-egy pont látszik. Minden sorban tizenhat byte szerepel (kivéve az esetleges első és utolsó csonka sort). A baloldalon, a nyolcadik és a kilencedik byte közé egy kötőjelet tesz a DEBUG.

A példánkban a DS regiszter által meghatározott szegmens hexadecimális 100-tól 17F-ig terjedő relatív címeinek adattartalmát íratjuk ki. Az alábbi három parancs ugyanezt a feladatot hajtja végre:

```
D
D DS:100 17F
D 100 L 80
```

Ha a tárképet nyomtatásban is látni akarjuk, használjuk a <Shift-PrtSc>, a <Ctrl-P> vagy a <Ctrl-PrtSc> billentyűkombinációkat.

9.2.2 R (register) - regiszterek kiíratása, megváltoztatása

Az R (regiszter) parancs lehetővé teszi a mikroprocesszor regisztereinek és flag-jeinek (jelzőbitjeinek) a megtekintését és módosítását. A parancs általános formája:

R [*regiszternév*]

ahol a *regiszternév* az alábbi nevek egyike lehet:

AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, IP, PC, F.

Az **IP** (Instruction Pointer) és a **PC** (Program Counter) rövidítések ugyanarra a regiszterre utalnak. Az **F** rövidítés a flag-eket (jelzőbiteket) jelenti.

Ha az R parancsot paraméter nélkül adjuk ki, akkor a DEBUG megjeleníti a képernyőn a regiszterek és a flag-ek tartalmát, valamint a következő végrehajtandó utasítás kódját és mnemonikus alakját. A regiszterek (AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, IP) tartalma hexadecimálisan íródik ki. Az egyes flag-ek értéke kétbetűs rövidítések formájában jelenik meg. A rövidítések magyarázatát az alábbi táblázat tartalmazza:

Flag neve	**Beállítva (1)**	**Törölve (0)**
Túlcsordulás (overflow) igen/nem	OV	NV
Irány (direction) csökkenő/növekvő	DN	UP
Megszakítás (interrupt) engedélyezve/letiltva	EI	DI
Előjel (sign) negatív/pozitív	NG	PL
Nulla (zero) igen/nem	ZR	NZ
Belső átvitel (auxiliary carry) igen/nem	AC	NA
Paritás (parity) páros/páratlan (even/odd)	PE	PO
Átvitel (carry) igen/nem (yes/no)	CY	NC

A következő végrehajtandó utasításról a DEBUG az alábbi adatokat közli:

- az utasítás kezdőcímét (ezt a CS:IP regiszterpár tartalmazza),
- hexadecimális kódját és
- visszafordított (disassemblált) alakját (azaz mnemonikját és operandusait).

Ha az R parancsban valamelyik regiszter nevét is megadtuk, akkor a DEBUG csak a megnevezett regiszter tartalmát írja ki (hexadecimálisan). Ezután - a következő sorban - egy kettőspont jelenik meg, amelyik mellé a regiszter új értékét írhatjuk be (egy maximum négyjegyű hexadecimális szám formájában). Ha ehelyett csak az < ENTER > -t nyomjuk le, akkor a regiszter tartalma nem változik.

Ha az R parancsban az F paramétert használjuk, akkor a DEBUG csak a flag-ek értékét jeleníti meg, a fenti táblázat rövidítéseivel. A kódok mellett egy kötőjelet látunk, emellé írhatjuk a megváltoztatni kívánt flag-ek értékeit (tetszőleges sorrendben, szintén a fenti rövidítéseket használva). A flag-ek kódjai között nem kötelező szóközt hagyni. Ha változtatás helyett csak az < ENTER > -t nyomjuk le, a jelzőbitek tartalma nem módosul.

Nézzünk meg néhány példát az R parancs használatára:

```
R
```

A képernyőn megjelenik a regiszterek tartalma és a flag-ek értéke, valamint a soron következő utasítás.

```
R DX
```

A képernyőn a DX regiszter tartalmát látjuk, például ezt:

```
DX 14D2
:1C2
```

Az alábbi példában a flag-eket vizsgáljuk meg:

```
R F
```

A képernyőn ezt látjuk:

```
OV UP EI PL NZ AC PE NC-CY NV
```

9.2.3 F (fill) - memória feltöltése

Az F (fill, feltöltés) parancs segítségével tetszőleges adattal tölthetjük fel a RAM memória kijelölt részét. A parancs általános alakja:

F *intervallum adatlista*

Az *intervallum* és az *adatlista* fogalmát a 9.2 pont bevezető részében ismertettük. Ha az *intervallum* kezdőcíme nem jelöl ki szegmenst, akkor a DS regiszter tartalma határozza meg a szegmens címét. A parancs hatására a DEBUG feltölti a kijelölt tartományt az *adatlistában* felsorolt adatokkal.

Ha az *adatlista* több byte-ot tartalmaz, mint az intervallum, akkor a felesleges byte-ok nem kerülnek be a memóriába. Ha az intervallum hosszabb az *adatlistá*nál, akkor az *adatlista* elemei mindaddig ismétlődnek a tárban, ameddig az intervallum végéhez nem érünk.

Az alábbi példában a DS által meghatározott szegmens hexadecimális 200-tól 2FF-ig terjedő részét töltjük föl nullákkal:

```
F 200 2FF 0
```

A következő parancs hatására a kijelölt intervallum (10000:0-tól 5F-ig) egy hat-byte-os listával töltődik fel. A lista elemei tizenhatszor ismétlődnek a tárban:

```
F 10000:0 L 60 2 F1 'EZ' 10 0
```

A memóriába a string karaktereinek ASCII-kódja kerül be hexadecimális formában, a lista tehát a következő byte-okból áll: 02, F1, 45, 5A, 10, 00.

9.2.4 E (enter) - memóriatartalom megváltoztatása

Az E (enter, adatbevitel) parancs kiadásával lehet a RAM memória tetszőleges byte-jait megváltoztatni. A parancs általános alakja:

E *cím* [*adatlista*]

A *cím* és az *adatlista* specifikációját a 9.2 pont bevezető részében ismertettük. Ha a *cím* csak egy relatív címből (offset-ből) áll, akkor a DS regiszter tartalma határozza meg a szegmens kezdőcímét.

Ha a parancsban nem adtunk meg *adatlistát*, akkor interaktív (párbeszédes) üzemmódban módosíthatjuk a tár tartalmát. A parancs hatására a DEBUG kiírja a sor elejére a megadott címet, és mellé az ott található byte hexadecimális értékét. A szám után egy pont jelenik meg. Ezután rajtunk van a sor, és a következő lehetőségek közül választhatunk:

- A szóköz-billentyű lenyomásával változatlanul hagyjuk a kiírt byte-ot. Ekkor a DEBUG a következő byte értékét mutatja.
- Új értéket adunk a megjelenített byte-nak, egy maximum kétjegyű hexadecimális szám beírásával. Ezután a szóközbillentyűvel léphetünk a következő, vagy a kötőjellel az előző byte-ra.
- A kötőjel lenyomására a DEBUG az előző cím tartalmát mutatja meg (új sorban). Ezt is módosíthatjuk.
- Ha lenyomjuk az < ENTER > -t, kilépünk az E parancsból.

Ha az E parancsban egy *adatlistát* is megadtunk, akkor a lista egyes byte-jainak tartalma átmásolódik a memóriába, az előírt címtől kezdve.

Az első példában az interaktív üzemmódot mutatjuk be:

```
E ES:2A0
```

A képernyőn a következőket látjuk:

```
0621:02A0 1F.
```

A megjelenített byte tartalmát 5B-re módosítjuk, majd a szóköz-billentyű lenyomásával tovább lépünk:

```
0621/02A0 1F.5B 41.
```

A 2A1-en elhelyezkedő byte-ot nem változtatjuk meg, hanem a szóköz-billentyűvel a következő byte-ra térünk át:

```
0621:02A0 1F.5B 41.  4A.
```

A 2A2-es byte-ot 4A-ról 00-ra írjuk át, majd az <ENTER>-rel kilépünk az E parancsból:

```
0621:02A0 1F.5B 41.  4A.00
```

Az alábbi példában a DS regiszter által kijelölt szegmens 100-tól 106-ig terjedő területére írjuk be az adatlista elemeit:

```
E 100 "PC DOS",0
```

9.2.5 C (compare) - memóriaterületek összehasonlítása

A C (compare, összehasonlítás) parancs byte-onként összehasonlítja a tár kijelölt területeit és jelzi a különbségeket. A parancs általános alakja:

C *intervallum cím*

Az *intervallum* és a *cím* fogalmát a 9.2 pont bevezető részében ismertettük. Ha a cím (vagy az intervallum kezdőcíme) csak egy relatív címből (offset-ből) áll, akkor a DEBUG a DS regiszter tartalmát tekinti a hiányosan specifikált szegmens kezdőcímének.

Az első memóriaterületet az intervallum jelöli ki, a másodiknak csak a kezdőcímére van szükség, mert az összehasonlítandó byte-ok számát az első terület hossza határozza meg.

Az eltéréseket a DEBUG az alábbi formában jelzi a képernyőn:

```
cím1 byte1 byte2 cím2
```

ahol cím1 és cím2 a két összehasonlított byte címe, byte1 és byte2 pedig azok tartalma.

A példában a CS:100-tól 14F-ig terjedő byte-okat hasonlítjuk össze a DS:100-tól kezdődő byte-okkal. Az intervallum megadásánál az L paramétert használjuk (a hossz kijelölésére):

```
C CS:100 L 50 DS:100
```

A DEBUG - a COMP paranccsal ellentétben - az összes eltérő byte-ot megmutatja. Ha a COMP parancs nem megfelelő két file összehasonlítására (pl. eltérőek a hosszak, ill. 10-nél több eltérés van és ezeket is látni akarjuk) a következőket tehetjük:

- behívjuk a DEBUG programot,
- betöltjük a memóriába az első file-t,
- betöltjük a második file-t,

- összehasonlítjuk őket a C (compare) paranccsal.

Példa: tegyük fel, hogy mindkét file mérete 64 Kbyte-nál kisebb, akkor az összehasonlítás a következőképpen mehet:

```
DEBUG
- F 2000:0 L 0 0
- F 3000:0 L 0 0
- N [d:][út]file1[.kit]
- L 2000:0
- N [d:][út]file2[.kit]
- L 3000:0
- C 2000:0 L 0 3000:0
```

Ha túl sok az eltérés, a listát a képernyőn megállíthatjuk <Ctrl-S>-sel, ill. nyomtatóra is kiírathatjuk, ha a C parancs előtt <Ctrl-Prtsc>-t nyomunk le.

Megjegyzés: amennyiben .COM vagy .EXE file-okat akarunk így összehasonlítani, előtte át kell nevezni azokat más kiterjesztésre!

9.2.6 M (move) - memóriaterület átmásolása

Az M (move, mozgatás) parancs átmásolja a memória kijelölt területét a megadott címre. (Az eredeti tárterület eközben nem változik, kivéve, ha a másolás cél-területe átfedi az eredeti intervallumot.) A parancs általános alakja:

M *intervallum cím*

Az *intervallum* és a *cím* megadási módjait a 9.2 pont bevezető részében ismertettük. Ha a cím (vagy az intervallum kezdőcíme) csak egy relatív címből (offset-ből) áll, akkor a DEBUG a DS regiszter tartalmát tekinti a meg nem határozott szegmens kezdőcímének.

9.2.7 S - karaktersorozat keresése

Az S (search, keresés) parancs adott memóriatartományban egy adott karaktersorozat összes előfordulási helyét megkeresi. A parancs alakja a következő:

S *intervallum adatlista*

Az *intervallum* és az *adatlista* fogalmát a 9.2 pont bevezető részében ismertettük. Ha az *intervallumban* nem adjuk meg a szegmens értékét, a program a DS regiszterben levő értéket használja. Például:

```
S CS:100 1500 "HELLO"0
```

parancs az aktuális programszegmens 100-1500 címei közti részt nézi végig a "HELLO"0 karaktersorozat után kutatva. Minden egyes előfordulás után kiírja a "H" betű helyét. Ha egyszer sem fordul elő az adott intervallumban, akkor semmilyen üzenetet sem kapunk.

9.2.8 Q - kilépés a programból

A Q (quit) parancs kiadása után a DEBUG befejezi működését és visszaadja a vezérlést a DOS-nak (vagy annak a programnak, amelyik a DEBUG-ot elindította). Ha a vizsgált programra szükségünk van, azt a W paranccsal a Q kiadása előtt el kell menteni.

9.2.9 A (assemble) - assemblálás

Az A (assemble, assemblálás) parancs segítségével assembly nyelvű utasításokat írhatunk. A DEBUG előállítja az utasítások kódját, és elhelyezi a memóriában. A parancs általános alakja:

A [*cím*]

A cím specifikálási módját a 9.2 pont bevezető részében ismertettük. Ha a címet nem írjuk ki és az első A parancsnál tartunk, akkor az assemblálás a CS:IP0 címen kezdődik. A későbbiekben, ha nem adunk meg címet, akkor az előző A parancs által tárolt utasítás utáni byte-nál folytatódik a kódgenerálás. Ezt az alapértelmezést bármikor módosíthatjuk, ha az A parancsban a konkrét címet is megadjuk.

A parancs kiadása után a DEBUG megjeleníti a képernyőn az aktuális kezdőcímet. Akkor beírhatunk egy assembly nyelvű utasítást, amely a 8088-as (vagy 8086-os) processzor, illetve a 8087-es segédprocesszor utasításkészletébe tartozik. Ha szintaktikai hibát vétettünk, akkor az "Error" (hiba) üzenetet kapjuk, és a DEBUG újra kiírja az aktuális címet.

Az egyes assembly utasítások beírása után nyomjuk le az <ENTER> billentyűt. Ekkor az utasítás kódja bekerül a tárba, és a DEBUG a következő memóriacímet mutatja. Ha nem kívánunk több utasítást kiadni, a cím után közvetlenül nyomjuk le az <ENTER>-t, így térhetünk vissza a DEBUG parancs-szintjére.

Az assembly utasítások beírásánál a következő szabályokat kell betartani:

- A numerikus értékeket csak hexadecimálisan írhatjuk, maximum négy számjeggyel.
- Az ismétlő (REP...), szegmenskijelölő (CS:, DS:, ES:, SS: stb.) prefixeket a hozzájuk tartozó utasítás elé kell írni (azzal egy sorba, vagy az előző sorba).
- A string-kezelő utasításokban közölni kell a string méretét. A MOVS utasítást például MOVSB, illetve MOVSW formában kell írnunk, attól függően, hogy egy byte-ot vagy egy szót (word) akarunk mozgatni.
- A távoli visszatérés (far return) mnemonikja: RETF.
- A DEBUG az ugró utasításokban megadott címek elhelyezkedése alapján dönti el, hogy rövid (2 byte-os), közeli (3 byte-os) vagy távoli (5 byte-os) ugró utasítást generáljon-e. Ezt az automatizmust kikapcsolhatjuk, ha a NEAR (közeli), illetve a FAR (távoli) prefixet írjuk a cím elé. A NEAR szót NE-re lehet rövidíteni. Példák:

```
0A27:0100 JMP 12C          ; rövid ugrás (2 byte-os)
0A27:0100 JMP NEAR 12C     ; közeli ugrás (3 byte-os)
0A27:0100 JMP FAR 12C      ; távoli ugrás (5 byte-os)
```

- A DEBUG nem tudja megkülönböztetni, hogy az egyes operandusok byte- vagy szó- (word) hosszúságú memóriaterületekre hivatkoznak-e. Ezért - ahol mindkét típusú hivatkozás megengedett - használjuk az operandusok előtt a BYTE PTR (rövidítve: BY), illetve a WORD PTR (rövidítve: WO) prefixeket.

Példák:

```
INC WORD PTR [200]
NOT BY [DI]
```

- A DEBUG mini-assemblere a közvetlen (immediate) operandusoknál nem vár el semmilyen megkülönböztető jelet. Így a

  ```
  MOV BH,1A
  ```

  utasítás hatására a BH regiszterbe a hexadecimális 1A közvetlen érték kerül. A tárcímekre való hivatkozást viszont szögletes zárójelek közé kell tennünk:

  ```
  MOV BH,[1A]
  ```

  Ennél az utasításnál a BH-ba a 001A memóriacím tartalma töltődik be.

- A DEBUG megengedi a DB (define byte, byte definiálása) és a DW (define word, szó definiálása) parancsok használatát, amelyek helyet foglalnak a tárban, és ott helyezik el a megadott adatlista elemeit. Példák:

  ```
  DB 2B,0,'Szöveg',FF
  DW 12A4,8A6
  ```

- Az indirekt címzési módszereket a megszokott módon alkalmazhatjuk:

  ```
  ADC AX,2A[BP + 3].[SI-1]
  POP [SI]
  PUSH [BP + DI]
  ```

- Egyes utasításokra az assemblerek két szinonim mnemonikot is alkalmaznak (például a JL utasítás megegyezik a JNGE-vel, a LOOPNE a LOOPNZ-vel). A DEBUG e szinonimákat is elfogadja.

- A 8087-es aritmetikai segédprocesszor utasításai előtt adjuk meg a WAIT vagy a FWAIT prefixet. Például:

  ```
  FWAIT FADD ST,ST(3)
  ```

Nézzünk meg végül egy rövid példát az A parancs használatára. Az assemblálás kezdőcíme legyen hexadecimális 1000. Írjuk be:

```
A 1000
```

A baloldalon látható címeket a DEBUG írja ki, az utasítások begépelése a mi feladatunk. Az utolsó sorban (a 07F1:1004-es cím mellett) csak az < ENTER > -t nyomtuk le, így léptünk ki az A parancsból:

```
07F1:1000 POP DX
07F1:1001 MOV AX,DX
```

```
07F1:1003 RET
07F1:1004
```

9.2.10 U (unassemble) - disassemblálás

Az U (unassemble, disassemblálás) parancs a memóriában található gépi kódú utasításokat assembly utasítások formájában jeleníti meg a képernyőn. A parancs általános alakja:

```
U [cím]
```

vagy

```
U [intervallum]
```

A *cím* és az *intervallum* megadásának módjait a 9.2 pont bevezető részében ismertettük. Ha a parancsot paraméter nélkül adjuk ki, akkor a legutóbbi U parancs által kiírt utasítás utáni byte-tól kezdődik a disassemblálás (így lehet egy hosszabb tárterületet a legegyszerűbben áttekinteni). Ha a DEBUG indítása óta most adtunk ki először U parancsot, akkor a kezdőcím a CS:IP0 lesz. Egyébként is, ha az *intervallum*ban vagy a *cím*ben nem nevezünk meg szegmensregisztert (vagy szegmenscímet), akkor a CS regiszter által mutatott szegmens tartalmát dolgozza föl az U parancs.

Ha a parancs második formáját alkalmazzuk (*intervallum* megadásával), akkor a DEBUG a kijelölt tárterületet disassemblálja. Ha csak egy *cím*et írtunk be, vagy az alapértelmezés szerinti címet használjuk, akkor a kezdőcímtől indulva harminckét byte disassemblálása történik meg. Az utasítások változó hossza miatt a program esetenként néhány byte-tal többet is feldolgoz az előírtnál (ha az utolsó utasítás túlnyúlik az intervallum határán).

A képernyőn a következő adatok jelennek meg: az utasítások címe, hexadecimális kódja és assembly nyelvű alakja. Az

```
U 100 L 6
```

parancsra például az alábbi választ kaphatjuk:

```
07D2:0100 1E       PUSH DS
07D2:0101 2BC0      SUB AX,AX
07D2:0103 B402     MOV AH,2
07D2:0105 206472    AND [SI + 72],AH
```

Ugyanezt az eredményt kaptuk volna az L 7 vagy az L 8 hosszak megadásával, vagy az

```
U 100 107
```

parancs kiadásakor is.

Ha a képernyőre nem férnek ki a DEBUG által kiírt adatok, a listázást a <Ctrl-NumLock> billentyűk lenyomásával félbeszakíthatjuk, majd bármely más billentyűvel újraindíthatjuk.

9.2.11 G (go) - program futtatása

A G (go, futtatás) parancs a memóriában található gépi kódú programot indítja el. A parancs általános alakja:

G [=*cím*][*cím* [*cím*]...]

A G parancs kiadásakor a vezérlés az egyenlőségjel után írt *cím*re adódik át (az alapértelmezés szerinti szegmens a CS regiszter által meghatározott kódszegmens). Ha ezt a címet nem adtuk meg, akkor (szintén a kódszegmensben) az utasításmutató (IP) regiszterben található címnél kezdődik a futtatás.

A G parancsban egyenlőségjelek nélkül írt *cím*ek (legfeljebb tízet sorolhatunk föl) a megszakítási pontokat (breakpoint-okat) jelölik ki. Ha a megszakítási pontoknál nem adtunk meg szegmenst, akkor a CS szegmens lesz az alapértelmezés. Amikor a program ezekhez a pontokhoz ér, a futás félbeszakad, és megvizsgálhatjuk a regiszterek vagy a memória tartalmát. Szétágazó program esetében több megszakítási pontot érdemes kijelölni az egyes ágakon, így megálláskor látni lehet, hogy a program melyik ága futott. Ez a lehetőség a programok tesztelésénél fontos szerepet játszik.

A megszakítási pontokon a DEBUG a program indítása előtt kicseréli az eredeti kódot egy speciális megszakító utasítás kódjával. Amikor a végrehajtás során egy megszakítási ponthoz érünk, a DEBUG visszaállítja valamennyi megszakítási ponton az eredeti utasításkódokat, majd a képernyőn - az R parancsnál leírt formában - megjeleníti a regiszterek és a flag-ek tartalmát, valamint a soron következő utasítást. Vigyázat: ha a program egyik megszakítási ponthoz sem ér el, akkor az eredeti utasítások visszaállítása nem történik meg!

A megszakítási pont elérése után egyszerűen lehet továbbfuttatni a programot: a G parancsban ne adjunk meg egyenlőségjeles címet(megszakítási pontokat kijelölhetünk). Az alapértelmezés szerinti CS:IP kezdőcím pontosan a megszakítási utasításra mutat.

Ha nem definiáltunk megszakítási pontokat (vagy a végrehajtás során ezekhez nem jutottunk el) és a program a futása végén visszaadja a vezérlést a DEBUG-nak, akkor a "Program terminated normally" (a program futása normálisan fejeződött be) üzenetet kapjuk. Ha a programot ezután ismét futtatni akarjuk, általában újra be kell töltenünk a lemezről (lásd az N és az L parancsokat).

A G parancs kiadásakor a veremben legalább hat byte szabad területnek kell lennie, különben a futtatás nem lehetséges.

Az első példában a már korábban betöltött programunkat futtatjuk le. A program kezdőcíme: 2F3. Megszakítási pontokat nem jelölünk ki:

```
G =2F3
```

A második példában a program az (alapértelmezés szerinti) 100-as címen kezdődik. Két megszakítási pontot definiálunk, mert kíváncsiak vagyunk a regiszterek és a tár tartalmára, amikor a program az 54A, illetve az 101C címekhez ér:

```
G 101C 54A
```

A megszakítási pontok beírásának sorrendje közömbös. Tegyük föl, hogy a program eljutott az 54A címhez. Ott a végrehajtás félbeszakad, a DEBUG visszaállítja mindkét megszakítási ponton az eredeti utasításkódokat, majd megjeleníti a regiszterek és a flag-ek tartalmát, valamint a következő utasítást.

9.2.12 T (trace) - lépésenkénti végrehajtás

A T (trace, nyomkövetés) parancs a memóriában található gépi kódú programot futtatja. A végrehajtás minden, a parancsban megadott számú utasítás végrehajtása után megszakad. A parancs általános alakja:

T [=*cím*] [*szám*]

ahol a futtatás kezdő*cím*ét az egyenlőségjel után lehet megadni (a 9.2 pont bevezető részében leírtak szerint), a *szám* helyére pedig egy maximum négyjegyű hexadecimális számot írhatunk (ez jelöli ki az egy menetben végrehajtandó utasítások számát).

Ha a kezdőcímet nem adjuk meg, a DEBUG a CS:IP regiszterpárban található címen kezdi el a futtatást. Ha csak egy relatív címet (offset-et) írunk be, akkor az a CS regiszter által meghatározott szegmensen belül érvényesül.

A szám megadása nem kötelező a T parancsban. Ha kihagyjuk, az alapértelmezése: 1.

A T parancs minden egyes utasítás végrehajtása után megjeleníti a képernyőn a regiszterek és a flag-ek tartalmát, valamint a következő utasítást (az R parancsnál ismertetett formában). Ha a kiírás túl gyors, bármikor félbeszakíthatjuk a <Ctrl-NumLock> billentyűkkel. A listázás bármelyik billentyű lenyomására folytatódik.

A T parancsban megadott számú utasítás végrehajtása után a DEBUG megáll és új parancsra vár. Ha ezután egy cím nélküli T parancsot adunk ki, akkor a program futtatása pontosan a félbeszakítás helyén folytatódik.

A példában a kódszegmens 20B címétől indítjuk a programot, és tíz (hexadecimálisan: A) utasítás végrehajtását kérjük:

```
T =20B A
```

A DEBUG mind a tíz utasítás végrehajtása után megjeleníti a regiszterek és a flag-ek tartalmát, valamint a soron következő utasítást. A program továbbfuttatásához az alábbi parancsot adjuk ki:

```
T A
```

Ezután - ha a programot tíz utasításonként tovább akarjuk futtatni - már csak az <F3> és az <ENTER> billentyűvel kell ismételni az előző parancsot.

9.2.13 P - utasítássorozat végrehajtása

A P (proceed) parancs hasonlóan működik, mint a T (trace) parancs, azzal a lényeges különbséggel, hogy bizonyos gépi kódú utasításokat egyetlen egynek tekint. Ezek a következők:

- szubrutinhívás;
- ciklusutasítás;
- megszakítás;
- ismételt string utasítás.

Az utasítás általános alakja:

P [=*cím*][*szám*]

Ha a memóriában a következő program található:

```
0100 CALL 1000
0103 INT 214
0106 JZ 3000
```

akkor a P2 utasítás hatására a programnak a CALL 1000 és CALL 2000 parancsai hajtódnak végre, függetlenül attól, hogy az 1000-nél kezdődő alprogram és a 21-es megszakítást kezelő alprogram valójában hány utasításból áll.

A P paranccsal lehetőségünk van a már letesztelt és ellenőrzött programrészek nyomkövetésének kikapcsolására. A P parancs csak a 3.xx verziókban használható.

9.2.14 N (name) - a file kijelölése

Az N (name, név) parancs segítségével hozhatunk létre file-vezérlő blokkokat (File Control Block), és specifikálhatjuk a file-neveket az L (load) és a W (write) parancsok előtt.

A parancs általános alakja:

N *filespec* [*filespec*]...

ahol a *filespec* a szokásos [*d:*][*út*]*file-név*[*.kit*] formában adható meg.

A DEBUG az első két file részére a DS:5C, illetve a DS:6C címeken hoz létre file-vezérlő blokkot. Az N parancs után írt valamennyi karakter (a file-specifikációkat is beleértve) átmásolódik a DS:81 címtől kezdve, a paraméter-területre.

A DS:80 címre a beírt karakterek száma kerül.

Ha a DEBUG indító parancsában már megneveztünk egy file-t, akkor a DS:5C címen elhelyezkedő file-vezérlő blokk az ott kijelölt file adatait tartalmazza. Az ezután kiadott L és W parancsok hatására is ugyanennek a file-nak a beolvasása, illetve kiírása fog megtörténni.

Ha a DEBUG indításakor nem adtunk meg file-nevet, vagy pedig nem az ott kijelölt file-t akarjuk olvasni (írni), akkor az L (ill. W) parancs előtt az N paranccsal kell az új file-t specifikálni.

A már betöltött programok esetében az N parancs segítségével tölthetjük föl a program file-vezérlő blokkjait és paraméterterületét a szükséges információkkal.

Tegyük fel, hogy a DEBUG indító parancsában nem adtunk meg file-nevet. Most viszont be akarjuk tölteni a PROBA nevű file-t a tárba. Az alábbi parancsokat kell kiadnunk:

```
N PROBA
L
```

A második példában feltételezzük, hogy a DEBUG indításakor már megtörtént a PROBA program betöltése. Most a program két file-vezérlő blokkját és paraméter-területét töltjük fel a kívánt adatokkal:

```
N ELSO.DAT MASODIK.TXT PARAM1 PARAM2
```

A DEBUG kialakítja a DS:5C és DS:6C című file-vezérlő blokkokat az ELSO.DAT és a MASODIK.TXT file-ok adataival (DS:5C-n eddig a PROBA file vezérlő blokkja helyezkedett el). A DS:80-on a paraméterekként megadott karakterek darabszáma (35, azaz hexadecimális 23) áll, DS:81-től kezdve pedig maguk a paraméterek (az N utáni szóköztől egészen a PARAM2 végéig).

9.2.15 L (load) - file (kijelölt szektorok) betöltése

Az L (load, betöltés) parancs betölti a memóriába a kijelölt file-t, illetve a megadott lemezes szektorok tartalmát. A parancs általános alakja:

L [*cím* [*meghajtó szektor szám*]]

ahol

cím	megadása a 9.2 pont bevezető részében leírtak szerint történhet,
meghajtó	helyébe egy egyjegyű számot lehet írni (a 0 szám az "A", az 1 a "B" lemezegységre utal stb.),
szektor	a beolvasni kívánt első szektor sorszáma (0-tól számozva). A 0. sorszám a 0. sávon a 0. fej által beolvasott 1. szektort jelenti. A sorszámozás a 0. sávon és a 0. fej pályáján folytatódik a 2., 3. stb. szektorokkal. Ezután - ugyanezen a sávon - az 1. fej alatt elhelyezkedő szektorok kerülnek sorra. Amikor a 0. sávon valamennyi fejet számba vettük, áttérünk az 1. sávra és így tovább.
szám	a beolvasandó szektorok száma, hexadecimálisan (maximális értéke: hexadecimális 80).

Ha az L parancsot paraméterek nélkül adjuk ki, vagy csak a címet írjuk be, akkor a DEBUG a DS:5C címen elhelyezkedő file-vezérlő blokkban (FCB-ben) specifikált file-t tölti be a tárba. Az FCB-t az N paranccsal lehet feltölteni, de ha ezt nem használtuk, akkor ott a DEBUG indító parancsában megnevezett file adatai szerepelnek.

A file-ok betöltése a DS:100 címtől történik, ha a parancsban ettől eltérő címet nem adtunk meg. A .COM és az .EXE kiterjesztésű file-okat a DEBUG - az általunk beírt címtől függetlenül is - a CS:100-ra tölti be. Ha az EXE file szerkesztésekor (LINK) a program magas címre való betöltését írtuk elő, akkor ezt az utasítást a DEBUG is betartja.

A file beolvasása után a BX és a CX regiszterek a file hosszát mutatják. Az .EXE és a .HEX kiterjesztésű file-ok betöltéséről, valamint a BX és a CX regiszterek tartalmáról a 9.1 pontban találunk bővebb leírást.

Ha az L parancsban lemezes szektorok beolvasását kérjük, akkor a betöltés kezdőcímét is meg kell adnunk. Amennyiben ez csak egy relatív címből (offset-ből) áll, a DEBUG a CS regiszter által mutatott szegmenst használja. A kijelölt szektorok tartalma ettől a címtől kezdve, folyamatosan tölti ki a memóriát.

Az alábbi példában a PACMAN.COM programot töltjük be a CS:100 címre. Ha a DEBUG indító parancsában a program neve már szerepelt (DEBUG PACMAN.COM), és N parancsokat nem használtunk, akkor elég ennyit írni:

```
L
```

vagy

```
L 100
```

Ha az indító parancsban a PACMAN.COM nevét nem adtuk meg, akkor az N parancsra is szükség van:

```
N PACMAN.COM
L
```

Az utolsó példában a "B" egységben levő kétoldalas floppy lemezről olvassuk be a tárba az 1. sáv, 0. oldal (azaz: 0. fej) valamennyi szektorának adatait. A sávon kilenc szektor található, az első szektor sorszáma: hexadecimális 12. A betöltés kezdőcíme legyen 1000:0:

```
L 1000:0 1 12 9
```

A szektorok egyenként 512 (hexa 200) byte-ot tartalmaznak, így a memóriában az 1000:0-tól az 1000:11FF-ig terjedő tartományt töltöttük fel adatokkal.

Megjegyzés:

Fontos tudni, hogy a floppy lemezekkel ellentétben a winchester-eknél a szektorszámozás a DOS partíció kezdetétől relatív (nem a winchester fizikai első cilinderének első sávjától), így DEBUG alól - az L paranccsal - nem tudjuk pl. a winchester partíció-táblát (0. cilinder 0. oldal 1. szektorát) beolvasni.

9.2.16 W (write) - file (kijelölt szektorok) kiírása

A W (write, kiírás) parancs kiírja a memória tartalmát egy file-ba, illetve a megadott lemezes szektorokba. A parancs általános alakja:

W [*cím* [*meghajtó szektor szám*]]

ahol a paraméterek jelentése megegyezik az L parancsnál leírtakkal (azzal az értelemszerű különbséggel, hogy itt nem beolvasásról, hanem kiírásról van szó).

Ha a W parancsot paraméterek nélkül adjuk ki, vagy csak a címet írjuk be, akkor a DEBUG a DS:5C címen elhelyezkedő file-vezérlő blokkban (FCB-ben) specifikált file-ba írja ki a tár tartalmát. Az FCB-t az N paranccsal lehet feltölteni, de ha ezt nem használtuk, akkor ott a DEBUG indító parancsában megnevezett file adatai szerepelnek.

A file-ba való kiírás a memória CS:100 címén kezdődik (kivéve, ha a parancsban ettől eltérő címet adtunk meg). A kiírt byte-ok számát a BX és a CX regiszterekben szereplő érték határozza meg (lásd a 9.1 pontban). Ha a DEBUG indítóparancsában előírtuk egy file betöltését, akkor a BX és a CX regiszterek kezdetben felveszik a file hosszának megfelelő értéket. Ha e regiszterek tartalmát nem módosítottuk, és a file hossza sem változott, akkor minden további nélkül kiadhatjuk a W parancsot. Egyéb esetben először korrigálnunk kell a BX és a CX regiszterek értékét.

Egy megszorítás: az .EXE és a .HEX kiterjesztésű file-okat nem írhatjuk vissza a W paranccsal, mivel a DEBUG ezeket a file-szerkezeteket nem tudja előállítani. Ha mégis ilyen kiterjesztésű file-t akarunk módosítani, akkor - a betöltés előtt - nevezzük át az adott file-t más kiterjesztésűre. A file-t ezután módosíthatjuk a DEBUG-gal, de futtatni (a G vagy a T paranccsal) nem tudjuk. A lemezre való kiíratás után visszaadhatjuk a file-nak az eredeti nevét.

Ha a W parancsban szektorokba való kiíratást kértünk, akkor a kiírandó tárterület kezdőcímét is meg kell adnunk. Amennyiben ez csak egy relatív címből (offset-ből) áll, a DEBUG a CS regiszter által mutatott szegmenst használja. A memória tartalma folyamatosan íródik ki ettől a címtől kezdve, amíg a megadott számú szektor fel nem töltődik.

Az alábbi példában az XYZ.DAT file-t írjuk vissza a lemezre. Ha a file-t a DEBUG indító parancsával, vagy a legutóbbi L paranccsal töltöttük be a CS:100 címre (továbbá a BX és a CX regiszterek tartalma, valamint a file hossza azóta nem változott meg), akkor elég az alábbi parancsot kiadni:

```
W
```

Ha a fenti feltételek nem teljesülnek, akkor először beírjuk az R paranccsal a BX és a CX regiszterekbe a file hosszát. Legyen a hossz hexadecimális 1000. Ekkor a BX-be nullát, a CX-be pedig az 1000 értéket írjuk. Ezután adjuk ki az alábbi parancsokat (a file-t a "C" meghajtóra írjuk):

```
N C:XYZ.DAT
W
```

A következő példában a kódszegmens 100-tól 4FF-ig terjedő részét írjuk ki konkrétan megnevezett szektorokba. (Vigyáznunk kell, hogy ezzel ne tegyük tönkre a lemezen tárolt file-okat!) A fenti memóriaterület hossza decimálisan 1024 byte, tehát két szektorban fér el. A kiírás kezdődjön az "A" egységben levő (kétoldalas) lemez második sávján, a nulladik oldalon, a harmadik szektorban:

```
W 100 0 26 2
```

A 100 tehát a kiírandó terület relatív kezdőcíme, a 0 a lemezegység száma (az "A" jelű floppy meghajtóé), a 26 az első írandó szektor sorszáma, a 2 pedig a kiírandó szektorok száma. A szektor sorszámát az alábbi táblázat magyarázza:

Sáv	Fej	Szektor	Szektor sorszáma
0	0	1	0
0	1	1	9
1	0	1	12
1	1	1	1B
2	0	1	24
2	0	3	26

9.2.17 H - hexadecimális műveletvégzés

A H (hexadecimális aritmetika) parancs segít a DEBUG parancsaihoz szükséges hexadecimális értékek kiszámításában. A parancs általános alakja:

H *szám szám*

ahol a két *szám* helyére legfeljebb négyjegyű, hexadecimális értékeket kell írni.

A H parancs hatására a DEBUG megjeleníti a képernyőn a két szám összegét és különbségét (szintén hexadecimális formában).

A példában a 2A31 címen kezdődő, 1BF hosszúságú string utolsó karakterének címét szeretnénk megtudni. A két szám összege már a string utáni byte-ra mutatna, ezért a string hosszából egyet levonunk:

```
H 2A31 1BE
```

A DEBUG válasza:

```
2BEF 2873
```

Az első érték adja a keresett címet, a második érték (a két szám különbsége) jelen esetben nem fontos számunkra.

9.2.18 I - input a kijelölt port-ról

Az I (input, adatbevitel) parancs beolvas egyetlen byte-ot a megadott című port-ról (a processzorhoz kapcsolt adatcsatornáról), és megjeleníti a képernyőn a beérkezett byte hexadecimális értékét. A parancs általános alakja:

I *portcím*

ahol a *portcím* egy maximum négyjegyű hexadecimális szám lehet (0-tól FFFF-ig).

A példában a 320-as című port-ról olvasunk, az eredményt (a 3A értéket) a DEBUG írja ki:

```
I 320
3A
```

9.2.19 O - output a kijelölt port-ra

Az O (output, adatkivitel) parancs kiírja a kijelölt port-ra (a processzorhoz kapcsolt adatcsatornára) a megadott egy-byte-os értéket. A parancs általános alakja:

O *portcím szám*

ahol a *portcím* egy maximum négyjegyű, a *szám* pedig egy maximum kétjegyű hexadecimális érték lehet.

A példában a 41A0 című port-ra írjuk ki a 2B értéket:

```
O 41A0 2B
```

10. Futtatható program előállítása. Programszerkesztés

A programszerkesztő - LINK - végrehajtható programot készít olyan tárgy-file-okból, amelyeket a rendszer különböző szintű fordítóprogramjai készítettek (MASM, C, PASCAL stb.). A szerkesztés eredménye egy program, amelyet a LINK egy végrehajtható program-file-ba (.EXE kiterjesztésűbe) másol. A felhasználó az így készített program futását a file-név PC DOS parancssorban való megadásával indíthatja.

A LINK a megadott tárgy-file-okból, valamint az esetleg szükséges könyvtári file-okból egy relokálható programot készít, amelynek helyigénye legfeljebb 1Mbyte lehet.

A továbbiakban részletesen leírjuk, hogyan vezérelhető a LINK, és hogyan készíti el a futtatható programot.

10.1 A LINK indítása

A LINK indítására és vezérlésére 3 lehetőség van:

- PC DOS parancs-sorral,
- a LINK kulcsszóval, majd a LINK által kiírt kérdésekre adott válaszokkal,
- egy előre összeállított válasz-file-lal.

A LINK működése bármikor leállítható a <Ctrl-Break> lenyomásával.

10.1.1 Programszerkesztés PC DOS parancs-sorral

A szerkesztést indító PC DOS parancs alakja:

LINK [*opciók*]*tárgy-file-ok*,[*futtatható file*],[*map-file*],[*könyvtár file-ok*]

Az "*opciók*" a LINK működését vezérlő feltételek. Ezek részletes leírása e fejezet 10.2. pontjában található.

A "*tárgy-file*" egy vagy több olyan tárgy-file neve, amelyből a programot szerkesztjük. Ezeket a tárgy-file-okat fordítóprogram - pl. makró assembler - készítette.

A "*futtatható file*" a LINK által elkészítendő program-file neve. Ha nem adjuk meg, akkor a LINK az első tárgy-file nevét feltételezi futtatható file-névként .EXE kiterjesztéssel.

A "*map-file*"-ba kerül az elkészült program térképének listája. Ha nem nevezünk meg ilyen file-t, vagy az opciók között nem adjuk meg a /MAP vagy a /LINE opciót, akkor nem készül map-file.

A "*könyvtár-file*" egy vagy több könyvtár-file neve, amely a programhoz szerkesztendő könyvtári rutinokat tartalmazza. Ha ilyenre nincs szükség, akkor a név helyébe pontosvesszőt kell írni.

A különböző típusú file-neveket vesszővel választjuk el egymástól. A vesszők kiírása akkor is kötelező, ha valamely file nevét nem adjuk meg. Ha egynél több tárgy-file vagy könyvtár-file neve szerepel, akkor a file-neveket legalább egy szóközzel vagy egy " + " jellel kell elválasztani egymástól. Az opciók bárhol lehetnek a parancs-sorban, csak azonos típusú file-nevek között nem.

A parancs-sorban szereplő file-nevekben nem kötelező megadni a kiterjesztéseket, mert a LINK feltételezi rendre a .OBJ (tárgy), .EXE (futtatható), .MAP (térkép) és .LIB (könyvtár) kiterjesztéseket.

Példák:

```
LINK ujit.OBJ, ujit.EXE, ujit.MAP, ujit.LIB
```

A parancs hatására a programszerkesztő az UJIT.OBJ tárgyfile-ból az UJIT.EXE végrehajtható program-file-t állítja elő az UJIT.LIB könyvtár-file-ban levő megfelelő rutinok hozzászerkesztésével. A program térképe az UJIT.MAP file-ba kerül.

```
LINK ujit,ujit,ujit,ujit
```

A parancs hatása ugyanaz, mint az előzőé, mert a LINK automatikusan hozzáteszi a file-nevekhez a .OBJ, .EXE, .MAP és .LIB kiterjeszést.

```
LINK ujit,ujit,ujit
```

A szerkesztés eredménye a LINK alapfeltételezései miatt ugyanaz, mint az előző két esetben.

```
LINK init + ujit,ujit,ujit,;
```

Két tárgy-file-ból (init.OBJ és ujit.OBJ) szerkeszti az ujit.EXE file-t könyvtár használata nélkül, és elkészíti az ujit.MAP file-t.

```
LINK x + y + init/PAUSE,,xy,\kat\elem
```

Az X.OBJ Y.OBJ és INIT.OBJ tárgy-file-okból, valamint a KAT könyvtárban levő ELEM.LIB alkönyvtár megfelelő rutinjaiból szerkeszti az X.EXE file-t és elkészíti az XY.MAP file-t. A /PAUSE opció hatására a végrehajtás felfüggesztődik, mielőtt a LINK elkezdené a futtatható program előállítását.

10.1.2 Programszerkesztés interaktív üzemmódban

Lehetőség van arra is, hogy a programszerkesztést párbeszédes üzemmódban hajtsuk végre. Ilyenkor az előző pontban ismertetett parancslista elemeit részenként adjuk meg.

A párbeszédes programszerkesztés lépései:

1. Begépeljük a szerkesztő program nevét:

```
LINK
```

és megnyomjuk az < ENTER > gombot. Ennek hatására a LINK megkérdezi a tárgymodulok nevét. A képernyőn a következő üzenet jelenik meg:

```
Object modules [.OBJ]
```

2. Az előbbi üzenetre válaszul beírjuk az összeszerkesztendő tárgy-file-ok nevét. Egynél több tárgy-file esetén a file-neveket szóközökkel vagy " + " jellel kell elválasztani egymástól. Ha a file-nevek nem férnek el egy sorban, akkor a sor utolsó karaktereként üssünk + jelet, majd nyomjuk meg az < ENTER > -t. Ekkor a LINK megkérdezi a többi tárgy-file nevét. A file-nevekben nem kötelező kiírni a .OBJ kiterjesztést, mert ez az alapfeltételezés.

Ha megadtuk az összes tárgy-file nevét, nyomjuk meg az < ENTER > -t! Ennek hatására a LINK a futtatható file nevét kéri:

Run File [file-név.EXE]

3. Beírjuk a futtatható file nevét és megnyomjuk az < ENTER > -t. Ha nem adunk meg file-nevet vagy kiterjesztést, akkor a LINK az alapértelmezést veszi figyelembe (első tárgymodul neve és .EXE). Ilyenkor elég csak az < ENTER > -t megnyomni.

A LINK ezután a map-file nevét kérdezi:

List File [NUL.MAP]

4. Ha készíteni akarunk map-file-t, akkor írjuk be a nevét és nyomjuk meg az < ENTER > -t! A .MAP kiterjesztést nem kötelező megadni, mert alapfeltételezés. Ha nem kérünk map-file-t, akkor elegendő csak az < ENTER > -t megnyomni.

A LINK következő kérdése a könyvtár neve:

Libraries [.LIB]

5. Valamennyi könyvtár nevét fel kell sorolni, amelyekben olyan rutinok vagy változók vannak, amelyekre a program hivatkozik, de nem deklarálja azokat. A nevek elválasztása szóközökkel vagy + jellel történik. A kiterjesztés alapértelmezése .LIB. A könyvtárneveket több sorban is megadhatjuk úgy, mint a tárgy-file-ok neveit.

Ha könyvtárhasználatra nincs szükség, akkor elég csak az < ENTER > -t leütni.

Ezek után a LINK előállítja a végrehajtható file-t.

Megjegyzés:

1. Ha valamely file másik könyvtárban vagy másik lemezen van, mint az aktuális alapértelmezés, akkor a file nevéhez meg kell adni a megfelelő elérési utat (path), vagy a lemez-azonosítót. Ha a LINK nem talál meg egy tárgy-file-t, akkor üzen és vár, amíg lemezt cserélünk, ha szükséges.
2. Minden sor végén lehetőség van opciók megadására.
3. Bármely üzenetváltás végződhet pontosvesszővel, melynek hatása: a LINK befejezi a párbeszédet, az elhagyott kérdésekre az alapfeltételezést alkalmazza, és elvégzi a programszerkesztést. Ügyeljünk arra, hogy legalább egy tárgy-file neve szerepeljen!

Példa:

A 10.1.1 pontban leírt utolsó példa párbeszédes formája a következő:

```
LINK
Object Modules [.OBJ]: x + y +
Object Modules [.OBJ]: init/PAUSE
Run File [x.EXE]:
List File [NUL.MAP]: xy
Libraries [.LIB]: \kat\elem
```

Az első sort a felhasználó írja be.

A 2. és 3. sor példa a file-nevek többsoros megadására. A 4. sorban láthatjuk a név alapfeltételezését (x), amely a végleges is, mert nem adtunk másikat.

10.1.3 Programszerkesztés előre összeállított válasz-file-lal

A LINK működéséhez szükséges információkat előre elhelyezhetjük egy file-ban, amelyet válasz-file-nak nevezünk. Ekkor a szerkesztés indítása:

LINK @válasz-file-név

A válasz-file-név tetszőleges lehet. A válasz-file tartalma:

tárgy-file-név
[*futtatható file-név*]
[*map-file-név*]
[*könyvtár-file-név*]

A file-ok megadási módjára az előzőekben leírt szabályok érvényesek. Minden file-típusnak külön sorban kell szerepelnie. Ha valamelyik file-típus helyett az alapfeltételezést használjuk, akkor a megfelelő sort üresen kell hagyni.

Indítás után a LINK elővészi a válasz-file-ból a megfelelő információkat, és automatikusan elvégzi a programszerkesztést. Szerkesztés közben a válasz-file tartalma folyamatosan megjelenik a képernyőn. Ha a LINK hibát talál a nevek olvasásakor, hibajelzést ad, és megvárja, amíg beírjuk a helyes választ.

Példa:

Az előző pontban bemutatott párbeszédes szerkesztési példának megfelelő válasz-file a következő:

```
x + y +
init/PAUSE

xy
\kat\elem
```

Figyeljük meg, hogy a harmadik sor nem látszik! Ha az xy ott volna, azt a LINK a RUN File kérdésére adott válasznak tekintené.

10.1.4 Könyvtárak elérési útjainak megadása

A könyvtárak elérési útjainak megadására két módszer lehetséges:

- A LINK parancsban lemezegységek és alkönyvtárak felsorolásával megadhatjuk azokat az elérési utakat, amelyeken haladni kell az adott név keresésekor.
- A második lehetőség az, hogy a LINK hívása előtt a PC DOS PATH parancsával értéket adunk a PATH környezetleíró változónak. Ez az érték egy vagy több elérési út.

Legfeljebb 16 elérési utat adhatunk meg. Az egyes elérési utakat pontosvesszővel választjuk el egymástól.

Egy elérési út leírása:

d:\alkönyvtár

Például:

A:\katall

Ha valamelyik hiányzik, akkor az aktuális lemezegység, illetve alkönyvtár a feltételezés.

Több elérési út megadása esetén a LINK a felírás sorrendjében balról jobbra haladva keresi a megfelelő könyvtárat.

Ha például a kt.LIB könyvtár elérési útjai:

A:\kat1;\kat2;B:

akkor először az "A" lemezegység \KAT1 alkönyvtárában keresi KT.LIB-et. Ha nem találja, az aktuális lemezegységen a \KAT2 alkönyvtár következik, s ha abban sincs, áttér a B: lemezegységre és az aktuális alkönyvtárban folytatja a keresést.

Példák:

1. LINK rend,,rend.MAP,A:\kat1\kt.LIB + dcl + B: + C:\kat2\

A LINK a KT.LIB könyvtárat csak az A: lemezegység KAT1 alkönyvtárában keresi, mert közvetlenül a könyvtárnév előtt adtuk meg az elérési utat. A DCL könyvtárat először az A: lemezegység aktuális könyvtárában keresi, majd a B: lemezegység aktuális alkönyvtárában, végül a C lemezegyseg KAT2 könyvtárában.

2. SET PATH = A:\kat1;B:;C:\kat2
   LINK rend,,rend.MAP, kt + dcl

Ebben a példában a LINK először az aktuális lemezegység aktuális alkönyvtárában keresi a KT és a DCL könyvtárat. Ha nem találja, akkor az A: egység KAT1 alkönyvtárában, a B: egység aktuális alkönyvtárában, majd a C: egység KAT2 alkönyvtárában folytatja a keresést.

10.1.5 A map-file szerkezete

A programról készített map-file lényegében a program főbb részeinek a program kezdetéhez relatív elhelyezkedéséről ad információt, valamint jelzi az esetleges hibákat.

Ha a LINK parancsban megadtuk a /MAP opciót, akkor kilistázza a public szimbólumokat és azok betöltési címeit is.

A map-file-ban levő szegmens-információ alakja:

```
START     STOP    LENGTH   NAME     CLASS
kkkkkH    vvvvH   lllllH   x...x    y...y
  .         .       .       .
  .         .       .       .
  .         .       .       .
```

A táblázat első két oszlopa a szegmensek kezdő és végcímét mutatja 20 bites hexadecimális alakban. Ezek a címek a load modul kezdetéhez relatívek.

A load modul feltételezett kezdőcíme 00000H. Ezekből a címekből képezi az operációs rendszer a valódi címeket a program betöltésekor.

A harmadik oszlop a szegmens byte-okban mért hosszát adja meg.

A negyedik oszlopban van a szegmens neve, az ötödikben a szegmens típusa (pl. CODE utasítások, DATA adatok).

A map-file-ban levő csoport-információ általános alakja:

```
 ORIGIN   GROUP
 nnnn:n   x...x
   .
   .
   .
```

Az első oszlop a csoport kezdetét, a második a nevét mutatja.

A LINK a program belépési pontjának címét a map-file végén helyezi el.

Ha a LINK parancssorában megadjuk a /MAP opciót, akkor a LINK a public szimbólumokról is elhelyez két táblázatot a map-file-ban. Az egyik táblázatban abc sorrendben, a másikban betöltési cím szerinti sorrendben szerepelnek a szimbólumok. A címek szegmens: relatív cím alakban kerülnek a map-file-ba. Ezek a load-modul kezdetéhez relatív címek.

Ha a LINK parancsban a /HIGH és /DSALLOCATION opció is szerepel és a program mérete az adatokkal együtt nem haladja meg a 64 Kbyte-ot, akkor az előzőekben említett táblázatokban az egyes szimbólumok mellett meglepően nagy címet találhatunk. Ez azt jelenti, hogy a szimbólum a program elé kerül.

10.1.6 VM.TMP - Ideiglenes lemez-file

A LINK munkaterülete a tárban van. Ha ez a terület programszerkesztés közben megtelik, akkor a LINK létrehoz egy ideiglenes lemez-file-t, amelynek neve az aktuális alkönyvtárban VM.TMP lesz. Ekkor a LINK a következő üzenetet küldi:

```
VM.TMP has been created.
Do not change diskette in drive <d:>
```

Az üzenet jelentése:

VM.TMP készült.
Ne cserélje a lemezt a < d > egységben!

Ez az üzenet a LINK működése végéig érvényes. A szerkesztés végrehajtása után az ideiglenes lemez-file automatikusan törlődik.

Megjegyzés:

Ha a LINK ideiglenes file-t hoz létre, az esetleg létező VM.TMP nevű file-t törli. Ezért ezt a nevet saját célra ne használjuk!

10.2 A LINK opciói

Az opciók segítségével vezérelhetjük a LINK működését. Minden opció "/" jellel kezdődik és a LINK parancsban bárhol előfordulhat, csak azonos típusú file-nevek között nem. A gyorsabb írás kedvéért minden opció neve rövidíthető.

Az opciók listája:

/PAUSE	megszakítja a szerkesztést
/MAP	kiírja a public szimbólumok listáját
/STACK	verem-méret
/HIGH	betöltés a lehetséges legnagyobb címtől
/DSALLOCATION	helyfoglalás adatcsoportnak
/LINE	forrás-sorok kezdőcímének kiírása
/X	az .EXE file által tartalmazott szegmensek száma
/O	a PASCAL és a FORTRAN fordítók 1. verziója által fordított modulok

10.2.1 PAUSE - Megszakítás a szerkesztés alatt

Alakja:

/PAUSE rövidítve: **/P**

Ezzel az opcióval megszakítható a LINK működése, mielőtt a végrehajtható file-t (.EXE) lemezre írja. Akkor szükséges, ha a kiírás előtt lemezt akarunk cserélni.

Ekkor a LINK üzenete:

```
About to generate .EXE file
Change disks <hit ENTER>
```

Az üzenet jelentése: Az .EXE file generálásához cserélje ki a lemez(eke)t és nyomja meg az ENTER billentyűt!

Az < ENTER > billentyű lenyomása után a LINK folytatja a munkát.

Ha VM.TMP nevű file van már a lemezen, akkor ezt a lemezt nem szabad cserélni.

Példa:

```
LINK rend/PAUSE,,,\kat\elem
```

A parancs hatása az, hogy a rend.EXE file lemezre írása előtt a LINK megáll, majd a lemezcsere után folytatja működését. A rend.EXE file lemezre írása után ismét megáll, lehetővé téve az eredeti lemez visszahelyezését.

10.2.2 MAP - Public szimbólumok táblájának előállítása

Alakja:

/MAP rövidítve: **/M**

Az opció hatására a LINK elkészíti a public szimbólumok táblázatát abc sorrendben, valamint cím szerint növekvő sorrendben (ld.10.1.5) és elhelyezi a map-file-ban.

Ha a LINK parancsban nem jelöltünk ki map-file-t, akkor a lista-file előtt megadott /MAP opció hatására automatikusan készül ilyen file, amelynek neve az első tárgy-file neve .MAP kiterjesztéssel.

Példa:

```
LINK rend/MAP;
```

ami egyenértékű a

```
LINK rend.OBJ,rend.EXE,rend.MAP/MAP,;
```

paranccsal.

10.2.3 STACK - A verem-méret meghatározása

Alakja:

/STACK:*méret* rövidítve: /S:*méret*

Hatására a programhoz tartozó verem helyfoglalása "*méret*" byte lesz. Értéke

1< = méret < = 65535

lehet.

A méret decimális formában adható meg.

A LINK általában automatikusan kiszámítja a verem méretét a tárgymodulokban megadott verem-szegmensek méretei alapján. Ezt a számított értéket lehet felülbírálni a LINK parancs /S opciójával.

Ha nincs verem-szegmens a programunkban, akkor a LINK hibajelzést ad. Ezt elkerülendő, minden programban legalább egy vermet definiáljunk!

Megjegyzés:

Amennyiben programokból .COM file-t akarunk előállítani, nem szabad STACK szegmenst definiálnunk. Ilyenkor figyelmen kívül hagyhatjuk a programszerkesztő figyelmeztető üzenetét. (A .COM előállítására .EXE file-ból az EXE2BIN segédprogramot használhatjuk.)

Példa:

```
        LINK rend/S:512;
vagy más alakban
        LINK rend.OBJ/STACK:512,rend.EXE;
```

Mindkét parancs 512 byte helyet biztosít a verem számára a REND nevű programban (futtatáskor).

10.2.4 HIGH - A kezdőcím maximumra állítása

Az opció hatására a program kezdőcíme a lehető legnagyobb cím lesz a tár szabad területén.

Az opció megadási módja:

/HIGH rövidítve: **/H**

Így lehet a programot a szabad hely végében elhelyezni.

Ha nem adunk meg ilyen opciót, akkor a betöltés a szabad hely elején kezdődik.

Csak a /DSALLOCATION paraméterrel együtt használjuk.

10.2.5 DSALLOCATION - Helyfoglalás az adatcsoportnak

Az opció hatására a DGROUP nevű csoportot tartalmazó szegmensben a csoport a lehető legnagyobb címre kerül.

Az opció alakja:

/DSALLOCATION rövidítve: **/D**

A LINK alapértelmezésben az adatcsoport első byte-jának szegmens-relatív címét 0000H-nak adja meg, míg az opció hatására a csoport utolsó byte-jához az FFFFH relatív címet rendeli.

Ezt az opciót legtöbbször a /HIGH opcióval együtt használjuk, mert így biztosítani tudjuk, hogy a program előtti fel nem használt tárterülettel jobban gazdálkodjunk.

10.2.6 LINE - Forrás-sorok kezdőcímének kiírása

Az opció hatására a LINK program minden forrás-sorából képzett utasítás(ok) kezdőcímét a map-file-ba írja.

Az opció alakja:

/LINE rövidítve: **/L**

A LINK által így készített információt fel lehet használni magasabb szintű nyelven írt programok forrásszintű nyomkövetéssel való teszteléséhez.

Az opció megadásának hatására csak akkor képez a LINK utasítás-kezdőcímeket, ha:

- van map-file-név a LINK parancsban,
- a tárgy-file-ban van sorszámozás (csak a magasszintű nyelvek fordítói készítenek ilyet).

Ha nincs a tárgy file-ban sorszámozás, akkor a LINK figyelmen kívül hagyja a LINE opciót.

Megjegyzés:

Ha nem adtunk meg map-file-nevet,akkor a LINE a MAP opció funkcióját is ellátja.

10.2.7 X - Szegmensszám beállítása

Az opcióval megadhatjuk az .EXE file által tartalmazott szegmensek maximális számát. Ez az érték 1-1024 között változhat. Ha az egyik opciót nem használjuk, akkor a programszerkesztő a 256 értékkel dolgozik.

Az opció alakja:

/X:*n* ahol *n*: 0-1024 lehet

10.2.8 O - 'Régi' fordítók használata

Az O (old) opció lehetővé teszi a Microsoft Pascal és FORTRAN fordítók 1. verziójával fordított file-ok szerkesztését is.

Az opció alakja: /O

11. A rendszer konfigurálása

A DOS az indítás során annak a meghajtónak a főtartalomjegyzékében, ahonnan indítottuk, megkeresi a CONFIG.SYS nevű konfigurációs file-t. A file-ban speciális parancsok vannak. Ezeket a DOS értelmezi, és ezek alapján beállítja a rendszer működésének jellemzőit.

A konfigurációs file tartalmának meghatározásakor figyelembe kell venni:

- az adott számítógép konfigurációját (memória mérete, fix és floppy meghajtók száma stb.),
- az adott számítógépen milyen alkalmazó rendszerek futnak (pl. dBASE III, sok I/O művelet),
- az is lehetséges, hogy egy-egy alkalmazói rendszer futtatása előtt a DOS rendszert újraindítjuk egy új konfigurációs file-lal.

11.1 A konfigurációs file kezelése

A felhasználó új CONFIG.SYS file-t szerkesztőprogrammal (pl. EDLIN) vagy a COPY paranccsal hozhat létre. Utóbbi esetben a következők a teendők. A DOS prompt-ja után begépeljük:

```
A>COPY CON CONFIG.SYS
```

és <ENTER>-t adunk, majd beírjuk a konfigurációs parancsokat, mindegyik után <ENTER>-t ütve. A COPY CON parancsot az <F6>, majd az <ENTER> billentyűvel zárjuk, ennek hatására a file-t a rendszer megőrzi.

Az új file hatása csak a következő rendszerindítás után érvényesül.

11.2 A konfigurációs parancsok

11.2.1 BREAK - a Break billentyű figyelése

Verzió: 2.11 - 3.30

A parancs formája:

```
BREAK [ON|OFF]
```

A paraméterek jelentése:

A rendszer az ON megadása esetén a DOS minden funkciójának végrehajtása előtt ellenőrzi, hogy a <Ctrl-Break> billentyűt leütöttük-e.

Az OFF megadása esetén csak konzol I/O, nyomtatási vagy aszinkron kommunikációs adapteri (soros átviteli) művelet esetén ellenőrzi a <Ctrl-Break> billentyűt.

Megjegyzés:

1. A CONFIG.SYS file-ban megadott BREAK-kapcsoló állapotát a BREAK DOS paranccsal meg lehet változtatni.

2. Ha a CONFIG.SYS file-ban nem adunk meg BREAK parancsot, akkor az a BREAK = OFF magadással azonos.

11.2.2 BUFFERS - lemez pufferek száma

Verzió: 2.11 - 3.30
A parancs formája:

```
BUFFERS = xx
```

A paraméter jelentése: Az *xx* 1-99 közötti egész szám, amely a lemez-pufferek számát jelenti. Minden egyes puffer a DOS méretét 528 byte-tal növeli, ezért a felhasználói memóriaterület csökken.

Alapértelmezés: 2 puffer (IBM PC AT esetén 3 puffer)

Megjegyzés:

1. A pufferek optimális számát a következő információk alapján állapíthatja meg a felhasználó:
   - a leggyakrabban használt alkalmazások típusa,
   - a számítógép memóriájának mérete,
   - a rendszer működésének elemzése különböző puffer-számok esetén.
2. A pufferek száma fix lemez esetén legalább 3 legyen.
3. Legalább 512 Kbyte memóriával rendelkező gépen javasolt legalább 20 puffert megadni: BUFFERS = 20

11.2.3 COUNTRY - dátum, idő és egyéb formátum megadása

Verzió: 3.00 - 3.30
A parancs formája (DOS 3.00-3.20):

```
COUNTRY = xxx
```

DOS 3.30 esetén a COUNTRY alakja a következő:

```
COUNTRY = xxx[,yyy],[d:]file-név[.kit]
COUNTRY = xxx,[yyy]
```

A paraméter jelentése:

Az *xxx* 3 decimális jegyből álló szám, az adott ország telefon kódja. Kezdeti értéke 001, ez az US formátum beállításának felel meg.

A konfigurációs parancs a dátum és idő kijelzési formátumát, a pénznem jelét és a tizedes helyét kijelölő karaktert állítja be. A DOS 3.20 az alábbi országokat ismeri fel:

Ausztrália	061
Belgium	032

Kanada (francia)	002
Dánia	045
Finnország	358
Franciaország	033
NSZK	049
Olaszország	039
Izrael	972
Közép Kelet	785
Hollandia	031
Norvégia	047
Portugália	351
Spanyolország	034
Svédország	046
Svájc	041
Egyesült Királyság	044
Egyesült Államok	001

yyy az adott ország kódhoz tartozó két kódlap valamenyike.

[*d:*]*file-név*[*.kit*] az országspecifikus információk használati módjára (pl.dátum, idő kijelzése) utaló információt tartalmazza. Ha elhagyjuk, akkor a program a COUNTRY.SYS file-lal dolgozik.

11.2.4 DEVICE - periféria-kezelő rutinok

Verzió: 2.11 - 3.20
A parancs formája:

DEVICE = [*d:*][*út*]*file-név*[*.kit*][*paraméterek*]

A paranccsal egy periféria-kezelő rutint tartalmazó file-t adunk meg. A DOS rendszer beépített periféria-kezelő rutinjai a következő egységek kezelését végzik:

- képernyő,
- billentyűzet,
- nyomtató,
- floppy egység (160/180 Kbyte, 320/360 Kbyte, 1.2 Mbyte),
- fix lemez.

Ha a számítógép konfigurációja csak ezeket a perifériákat tartalmazza, akkor nem kell megadni a DEVICE konfigurációs parancsot.

A DOS 3.30 változata öt periféria-kezelő rutint használ. Ezek a következők:

ANSI.SYS
DISPLAY.SYS
DRIVER.SYS
PRINTER.SYS
VDISK.SYS

Az ANSI.SYS az ANSI szabvány szerinti képernyőkezelést biztosítja. A VDISK.SYS me-

memórialemezt hoz létre, csak a DOS 3.00 vagy későbbi változataihoz használható. A DRIVER.SYS fizikai és logikai lemezegységek egymáshoz rendelését végzi el, s lehetővé teszi külső fizikai egységek használatát is. Csak a DOS 3.20-tól használható! A DOS 3.30-as változatától a nyomtatóra és a képernyőre biztosítja a kód-lapozás használatát a PRINTER.SYS, illetve a KEYB.SYS periféria-kezelő. A továbbiakban részletesen ismertetjük ezek használatát.

11.2.4.1 A kibővített képernyő- és billentyűzet-kezelés (ANSI.SYS)

A 2.00 az első olyan DOS verzió, amely megengedi, hogy a felhasználói programok speciális jelsorozatok kiadásával (escape-szekvencia) vezérelhetik a képernyőt és a billentyűzetet.

Megjegyzés:

E lehetőség alkalmazásához szükséges, hogy a kibővített képernyő- és billentyűzet-kezelés a DOS rezidens részét alkossa.

Ez úgy érhető el, hogy a CONFIG.SYS file-ba beszerkesztjük (pl. EDLIN-nel) a következő utasítást:

```
DEVICE=ANSI.SYS
```

Ez behívja a memóriába - a rendszer újraindításakor - a szükséges kezelőprogramot. Arra figyelemmel kell lenni, hogy a DOS memóriaigénye az ANSI.SYS méretével megnövekszik.

A vezérlő parancsok formája:

ESC[*paraméterek feladatazonosító*

Figyelem! A [jel része a parancsnak, közvetlenül az ESC (decimális 27, hexadecimális 1B) után áll!

A paraméterek és a feladatazonosítók lehetséges értékeit a további alszakaszokban ismertetjük.

A fenti "parancs" nem DOS parancs az eddigi értelemben, nem adhatjuk ki a fent leírt formában, a DOS akkor szintaktikus hibát jelezne (Bad command or filename).

Az egész "parancssort" az ESC ASCII kódjának (hexadecimális 1B, decimális 27) figyelembevételével kell a képernyőre kiírni, vagy a DEBUG, vagy a BASIC segítségével.

e200	1B "[2;10H"
az adatszegmens 200-as címére beteszi a DEBUG	ezt az escape szekvenciát (itt a 2;10H a paraméter és feladatazonosító)

Ezután, ha ezt a címet a képernyőre küldjük, a kurzor a 2. sor 10-ik oszlopára áll. Ugyanezt BASIC-ből:

```
10 PRINT CHR$(27); "[2;10H"
```

Mint látjuk, a paramétereknek és a feladatazonosítónak egy összefüggő sztringben kell lennie.

A "paraméterek" numerikus értékek. Az egyes lehetőségek leírásánál annyi kettőskeresztet (#) fogunk megadni, ahány ilyen érték az adott lehetőségnél megadható. A numerikus értékek decimális pozitív egész számok.

Ha egy paramétert elhagyunk, vagy 0-nak adunk meg, a paraméter hallgatólagos értéke lesz érvényes.

A "feladatazonosító" az adott lehetőséget azonosítja.

A kibővített képernyő-kezelés

- a kurzor mozgatását,
- a képernyő törlését és
- üzemmódjának beállítását végzi.

1. **A kurzor állítása**: ESC[#;#H

A kurzort a paraméterekkel megadott pozícióra mozgatja. Az első érték a sor, a második az oszlop értékét adja. Az alapértelmezés 1;1. Ha paramétert nem adunk meg, akkor a kurzor alaphelyzetbe (HOME) áll.

2. **Kurzor fel**: ESC[#A

A # azt adja meg, hogy a kurzort hány sorral mozgassa felfelé. Alapértelmezés:1. A képernyő tartalma mozgatás közben nem változik. Ha a kurzor a legfelső sorban áll, a parancs hatástalan.

3. **Kurzor le**: ESC[#B

A # azt adja meg, hogy a kurzort hány sorral mozgassa lefelé. Alapértelmezés:1. A képernyő tartalma mozgatás közben nem változik. Ha a kurzor a legalsó sorban áll, a parancs hatástalan.

4. **Kurzor előre (jobbra)**: ESC[#C

A # azt adja meg, hogy a kurzort hány oszloppal mozgassa jobbra. Alapértelmezés:1. A képernyő tartalma mozgatás közben nem változik. Ha a kurzor a jobbszélső oszlopban van, a parancs hatástalan.

5. **Kurzor vissza (balra)**: ESC[#D

A # azt adja meg, hogy a kurzort hány oszloppal mozgassa balra. Alapértelmezés: 1. Mozgatás közben a képernyő tartalma nem változik. Ha a kurzor a balszélső oszlopban van, a parancs hatástalan.

6. **Kurzor vízszintes és függőleges mozgatása**: ESC[#;#f

A kurzort a paraméterekkel megadott pozícióra mozgatja. Az első a sor, a második az oszlop sorszámát adja meg. Ha nem adunk meg értéket, akkor a kurzor alaphelyzetbe (HOME) áll.

7. **Kurzor helyzetének kijelzése**: ESC[6n

Ez lehetővé teszi a kurzor helyzetének folyamatos kijelzését.

8. **Kurzor pillanatnyi helyzetének kijelzése a standard input periférián**: ESC[#;#R

A kurzor pillanatnyi helyzetét adja meg. a standard input periférián. Az első paraméter a pillanatnyi sor, a második az oszlop sorszámát adja meg.

9. **Kurzor pozíciójának megőrzése**: ESC[s

A kurzor pillanatnyi helyzetét a rendszer megjegyzi. Ezt a pozíciót a 10. paranccsal lehet visszaállítani.

10. **Kurzor pozíciójának visszaállítása**: ESC[u

A kurzort abba a pozícióba helyezi, amelyben ESC[s kiadásakor volt.

11. **Képernyő törlése**: ESC[2J

A teljes képernyő törlődik, a kurzor alapállapotba kerül.

12. **Törlés sor végéig**: ESC[K

A kurzor pozíciójától a sor végéig valamennyi jelet töröl, azt is, amelyen a kurzor áll.

13. **A grafikus attribútumok beállítása**: ESC[#;...;#m

értékei a következők lehetnek:

0	normál fekete-fehér kijelzés
1	dupla erősségű kijelzés (high intensity)
4	aláhúzás bekapcsolása (csak IBM monokróm képernyő esetén használható)
5	villogó kijelzés (blink on)
7	fordított kijelzés (reverse video on)
8	törlés bekapcsolása (a képernyőn nem látható)
30	az előtér színe fekete
31	az előtér színe piros
32	az előtér színe zöld
33	az előtér színe sárga
34	az előtér színe kék
35	az előtér színe viola
36	az előtér színe kobaltkék
37	az előtér színe fehér
40	a háttér színe fekete
41	a háttér színe piros
42	a háttér színe zöld
43	a háttér színe sárga
44	a háttér színe kék
45	a háttér színe viola
46	a háttér színe kobaltkék
47	a háttér színe fehér.

14. Képernyő beállítása: ESC[= #h

értéke a következő lehet:

0	40x25 fekete-fehér
1	40x25 színes
2	80x25 fekete-fehér
3	80x25 színes
4	320x200 színes grafikus
5	320x200 fekete-fehér grafikus
6	640x200 fekete-fehér grafikus
7	A "sor vége" beadása után (END OF LINE) új sort kezd (Line Feed).

15. Képernyő visszaállítása: ESC[= #l

értéke ugyanaz lehet, mint a 14. pontnál leírtak. Ha viszont az ESC[= 7l értéket adjuk meg, akkor a "sor vége" beadása után (END OF LINE) nem lép új sorba.

A DOS lehetőséget biztosít arra, hogy a tasztatúrán lévő jel-billentyűk jelentését újra definiáljuk, vagy egy billentyűhöz karaktersorozatot rendeljünk.

1. ESC[#;#;...;#p

A # egy ASCII kód. Az első # az átalakítandó kód, a többi pedig azoknak a jeleknek az ASCII kódja, amelyek akkor jelennek meg a képernyőn, ha a megadott billentyűt lenyomjuk.

2. ESC[#;"karaktersorozat";#p

Az első # által meghatározott ASCII kódhoz egy karaktersorozatot rendelünk.

Megjegyzés:

Ha az első # értéke 0 (nulla), akkor ezt még egy # követi. A két # egy kibővített ASCII kódot jelöl.

Példa:

1. A kis q billentyűhöz a nagy Q-t rendeljük hozzá:

```
ESC 113;81p
```

2. A <PF10> billentyűhöz a DIR karaktersorozatot rendeljük, majd ezt a "sor vége" kóddal le is zárjuk.

```
ESC 0;68;"DIR";13p
```

A 0;68 az <PF10> funkció-billentyűt jelenti, a 13 pedig a "sor vége"-kód.

11.2.4.2 DISPLAY.SYS - kódlapok használatának engedélyezése a képernyőn

A periféria-kezelő használata lehetővé teszi, hogy azokon az eszközökön, amelyekerre fizikailag alkalmasak, különböző kódlapokat használjunk. Ezek az eszközök: IBM LCD típusú képernyő, EGA grafikus adapterű gépek és az IBM PS/2 típusú mikroszámítógépek.

Verzió: 3.30

A parancs formája:

1. **DEVICE = [*d:*][*út*]DISPLAY.SYS CON[:] = (*típus*[,[*kp*][,*n*]])**
2. **DEVICE = [*d:*][*út*]DISPLAY.SYS CON[:] = (*típus*[,[*kp*][,(*n*,*m*)]])**

A paraméterek jelentése:

típus	a rendszerben levő grafikus kártya típusát adja meg. Lehetséges értéke: MONO, CGA, EGA vagy LCD. Ha IBM PS/2 típusú gépünk van, akkor is az EGA-t kell megadnunk, holott a két grafikus adapter funkcionálisan nem ekvivalens. Ilyen esetben a program automatikusan felismeri, hogy melyikről van szó.
kp	a szóbanforgó hardware által támogatott kódlap sorszáma. A szóbajöhető értékek: **437, 850, 860, 863** és **865**.
n	a kiegészítő kódlapok számát adja meg, n értéke a 0-12 tartományba kell, hogy essen. MONO és CGA típus esetén n értéke értelemszerűen 0.
m	a betűtípusok számát adja meg az adott képernyő-típuson belül. Ha m-et nem adjuk meg, akkor azt a típus függvényében választja meg a program. LCD esetén (karakterméret 8x8) m = 1. EGA (karakterméret 8x8 vagy 8x14) és IBM PS/2 (karakterméret 8x8 vagy 8x16) esetén m = 2
út	a DISPLAY.SYS file-t tartalmazó tartalomjegyzék.

Példa:

```
DEVICE = C:\DOS\DISPLAY.SYS CON: = (EGA,865,2)
```

11.2.4.3 A DRIVER.SYS használata

A DEVICE konfigurációs parancsban használhatjuk a DRIVER.SYS periféria-kezelő rutint, mely lehetővé teszi fizikai és logikai lemezegységek egymáshoz rendelését.

Verzió: 3.20-3.30

A parancs formája:

DEVICE = [*d:*][*út*]DRIVER.SYS /D:*ddd*[/T:*ttt*][/S:*ss*][/H:*hh*][/C][/N][/F:*f*]

A paraméterek jelentése:

/D	*ddd* a fizikai meghajtó számát adja. A fizikai meghajtó szám a 0-255 intervallumba eshet. A 0-ás az első belső, az 1-es a második belső hajlékonylemezes meghajtó. A 2-es meghajtó szám a harmadik hajlékonylemezes egységé, amelyik azonban már csak külső egység lehet. Az első fix lemezegység fizikai meghajtó száma 128, a másodiké 129 stb.
/T	*ttt* megadja az egy oldalon levő sávok (track) számát. Kezdeti értéke 80 sáv/oldal. (Lehetséges érték: 1-999.)
/S	*ss* megadja az egy sávon levő szektorok számát. Kezdeti értéke 9 szektor/sáv. (Lehetséges érték: 1-99.)

/H	*hh* a lemez író/olvasó fejeinek maximális száma. Kezdeti érték: 2. (Lehetséges érték: 1-99.)
/C	jelzi, hogy a lemezegység kezeli az ún. "changeline" opciót (érzékeli hogy kicseréltük a lemezt). Ennek csak IBM PC AT és kompatibilis gépek esetén van hatása.
/N	megadja, hogy az egység nem cserélhető. Ilyenek a fix lemezek.
/F	*f* megadja a periféria típusát. f lehetséges értékeit az alábbi táblázat tartalmazza:

f	egység
0	160/180 Kbyte
0	320/360 Kbyte
1	1.2 Mbyte
2	720 Kbyte
7	1.44 Mbyte

Kezdeti értéke 2.

Példa:

1. Gépünk (egy IBM PC XT) egy belső hajlékonylemezt és egy fix lemezt tartalmaz. Ha a géphez csatlakoztatott (külső) 720 Kbyte-os egységet szeretnénk használni, akkor a CONFIG.SYS file-ba az alábbi parancsot kell beírnunk:

```
DEVICE = DRIVER.SYS /D:2
```

Ebben az esetben a külső lemezegységet a D betűjellel használhatjuk. Ha ezt az egységet másolásra is akarjuk használni, akkor a parancsot kétszer kell a CONFIG.SYS file-ban elhelyezni:

```
DEVICE = DRIVER.SYS /D:2
DEVICE = DRIVER.SYS /D:2
```

Az első parancsra létrejön egy D, a másodikra egy E betűjelű lemezegység. Ezután kiadhatjuk pl. a DISKCOPY E: D: parancsot. Fizikailag mindkettő a harmadik (külső) hajlékonylemezes egységgel azonos.

Előfordul, hogy a második belső lemezegység (B:) helyére helyezik el a streamert. Ebben az esetben a DISKCOPY A: B: parancs értelemszerűen nem használható. Ebben az esetben ehelyett létrehozunk egy D: logikai egységet, amelyik ugyancsak a 0-ás belső fizikai egységet használja:

```
DEVICE = DRIVER.SYS /D:0 /T:40 /S:15 /H:2 /F:0
```

Ennek hatására a DISKCOPY A: D: parancsot használhatjuk.

11.2.4.4 PRINTER.SYS - kódlapok használatának engedélyezése a nyomtatásra

A periféria-kezelő lehetővé teszi, hogy azokon a nyomtatókon, amelyek erre fizikailag alkalmasak, különböző kódlapokat használjunk. Jelenleg két nyomtató típust támogat a periféria-kezelő, az IBM Proprinter Model 4201 és az IBM Quitwriter III Printer Model 5202 típusokat.

Verzió: 3.30

A parancs formája:

1. **DEVICE =** [*d:*][*út*]**PRINTER.SYS LPT#**[:] = (*típus*[,[*kp*][,*n*]])
2. **DEVICE =** [*d:*][*út*]**PRINTER.SYS LPT#**[:] = (*típus*[,[(*kp1,kp2*,...)][,*n*]])

A paraméterek jelentése:

út	a PRINTER.SYS file-t tartalmazó tartalomjegyzék.
#	a nyomtató sorszáma. A használható eszköznevek: LPT1, LPT2 és LPT3. Ennek megfelelően # értéke 1, 2 vagy 3 lehet. LPT1helyett írhatunk PRN-t is.
típus	a szóbanforgó nyomtató típusa: 4201 vagy 5202.
kp,kp1,kp2...	a használni kívánt, a hardware által támogatott kódlapok száma. Lehetséges értékei: **437, 850, 860, 863, 865**.
n	a még használható kódlapok száma. Maximális érték 12.

Példa:

DEVICE = PRINTER.SYS LPT1: = (4201,850,)

11.2.4.5 VDISK.SYS - memórialemez használata

A VDISK.SYS periféria-kezelővel memória lemezt hozhatunk létre. A memória lemez pontosan úgy használható, mint a rendes lemez, azzal a lényeges különbséggel, hogy a gép kikapcsolásakor ennek adatai elvesznek. Memórialemez létrehozásához az alábbi parancsot kell elhelyezni a CONFIG.SYS file-ban:

DEVICE = [*d:*][*út*]**VDISK.SYS**[*megj*][*bbb*][*megj*][*sss*][*megj*][*ddd*][/**E**[:*m*]]

Verzió: 3.00 - 3.30

A paraméterek jelentése:

[*d:*][*út*]	azt a tartalomjegyzéket adja meg, ahol a VDISK.SYS file található.
megj	tetszőleges karaktersorozat, melyet megjegyzésekre lehet használni.
[*bbb*]	megadja a memória lemez nagyságát Kbyte-okban. Kezdeti értéke 64 Kbyte. A periféria-kezelő ezt az értéket az alábbi szabályok szerint módosíthatja:
a)	Ha a memórialemez installálása után kevesebb, mint 64 K memória maradna, a rutin csökkenti a bbb értékét. Erről üzenetet kapunk.
b)	Ha a memóriában már eleve 64 K-nál kisebb hely van, akkor a rutin nem installálja a memória lemezt. Erről is üzenetet kapunk.
[*sss*]	a szektor méret byte-okban. A lehetséges értékek: 128, 256 vagy 512. Ha nem adjuk meg, vagy helytelen értéket adunk meg, akkor a rutin a 128 értéket használja.
[*ddd*]	a főtartalomjegyzék bejegyzéseinek száma. (2 és 512 között lehet)
[/**E**[:*m*]]	megadja, hogy a memória lemezt kibővített memória területen használjuk-e. Ezt az opciót természetesen csak akkor használhatjuk, ha a gépben van kibővített memória (1Mbyte felett). m adja meg, hogy egyszerre maximum hány szektort lehet átvinni a bővített memória és a memória között. m lehetséges értéke 1,2,3,4,5,6,7 és 8. Kezdeti érték 8.

Megjegyzés:

1. A főtartalomjegyzék bejegyzések számát is módosítja a rutin, úgy, hogy a legközelebbi szektorhatárig további bejegyzéseket foglal le.
2. Ha a memória lemez nagysága kevés ahhoz, hogy a "boot" szektort, a FAT-ot és a főtartalomjegyzéket és még két szektort tartalmazzon, akkor a rutin ennek megfelelően megnöveli a memória lemez méretét.
3. A memória lemeznek automatikusan generál a rutin egy kötetazonosítót.
4. A memória lemez nem formázható meg.
5. A memória lemezek számának és méretének csak a rendelkezésre álló memóriaterület szab határt.
6. A /E paraméter esetén a 19H megszakítást használja a rutin a bővített memória kezelésére.

Példa:

Tegyük fel, hogy gépünk egy belső hajlékonylemezes és egy fixlemezes egységet tartalmaz. A

```
DEVICE = VDISK.SYS 180 512 64
```

konfigurációs parancs segítségével egy 180 Kbyte nagyságú memória lemezt hozunk létre. Szektorméret 512, a főtartalomjegyzék bejegyzéseinek száma 64. A memória lemezre a D betűjellel hivatkozhatunk.

11.2.5 FCBS - maximálisan megnyitható FCB-k száma

A konfigurációs parancs segítségével megadhatjuk, hogy egyidejűleg maximum hány file kontroll blokkot (FCB-t) nyithat meg a DOS.

Verzió: 3.00 - 3.30
A parancs formája:

```
FCBS = m,n
```

A paraméterek jelentése:

m az egyidejűleg megnyitható FCB-k száma. Értéke 1 és 255 között változhat. Alapértelmezése: 4.

n azon FCB-k száma, amelyeket nem lehet automatikusan lezárni, amikor a DOS az m + 1-ik FCB-t kísérli megnyitni. A DOS, ha az m + 1-ik FCB-t akarjuk megnyitni, kénytelen valamelyik előző megnyitottat lezárni. n megadása esetén az első n megnyitott FCB nyitva is marad. Lehetséges értéke: 0-255. Kezdeti érték:0.

Megjegyzés:

1. m legalább akkora kell, hogy legyen, mint n.

2. Ha egy program több FCB-t használ ugyanarra a file-ra, azt a DOS csak egynek tekinti.

3. Az FCBS parancsnak csak akkor van hatása, ha a file megosztási környezetet a SHARE DOS paranccsal betöltöttük. Ha az m + 1-ik FCB-t akarjuk megnyitni, akkor a legrégebben használt FCB-t (de nem az első n közül!) lezárja a DOS. Előtte azonban

   ```
   FCB unavailable
   Abort? Retry? Ignore?
   ```

   üzenetet kapunk.

11.2.6 FILES - egyidejűleg megnyitható file-ok száma

Verzió: 3.00 - 3.30

A parancs formája:

```
FILES = xxx
```

A paraméter jelentése:

A FILES = xxx az egyidőben megnyitott file-ok maximális számát jelenti.

Az xxx 8-255 közötti egész szám. A DOS a file-ok kezelése során minden file-hoz egy file-leíró blokkot rendel. A DOS 8 db file-leíró blokkot a saját területén tud tárolni, tehát az alapértelmezés:8. Ha nem elégséges ez a terület, akkor a FILES érték megnövelésével lehet pótlólagos memória területet lefoglalni.

Megjegyzés:

1. Ha nem elégséges a terület a file-ok megnyitásához, akkor hibaüzenetet kapunk.
2. A FILES = xxx érték meghatározásánál nemcsak a felhasználói programok, hanem a rendszer-komponensek file-jait is figyelembe kell venni, így pl. dBASE III esetén a FILES értéke legalább 24 legyen.

11.2.7 LASTDRIVE - a legnagyobb lemezegység megadása

A konfigurációs parancs megadja a logikailag használható lemezegységek maximális számát.

Verzió: 3.00 - 3.30

A parancs formája:

```
LASTDRIVE = d
```

A paraméter jelentése:

d az utolsó, még logikailag használható lemezegység betűjele. Ha pl. 14 egységet akarunk használni, akkor a

LASTDRIVE = N

konfigurációs parancsot kell a CONFIG.SYS-ben elhelyezni. Ezután az A-N lemezazonosítókat használhatjuk. (Természetesen csak azokat, amelyekhez hozzá is rendeltünk egységet.)

11.2.8 SHELL - felhasználói parancs-processzor

A parancs lehetővé teszi, hogy a felhasználó saját parancs-processzorát töltse be a DOS rendszer indítása során.

Verzió: 2.11 - 3.30

A parancs formája:

SHELL = [*d:*][*út*]*file-név*[*.kit*][*paraméter1*][*paraméter2*]

A paraméterek jelentése:

Olyan file-t adunk meg, amely a DOS betöltése után a COMMAND.COM funkcióját veszi át.

Ily módon lehetőség van a parancs-processzor lecserélésére. A paraméterek jelentése a használt parancs-feldolgozótól függ.

11.2.9 STACKS - a DOS veremhasználatának módosítása

A konfigurációs parancs segítségével módosíthatjuk, hogy a DOS hány és mekkora vermet allokáljon egy-egy hardver megszakítás esetén.

Verzió: 3.20-3.30

A parancs formája:

STACKS = *n*,*s*

A paraméterek jelentése:

n	a használható vermek száma. Alapértelmezése 9. Ezt az értéket 8 és 64 közé állíthatjuk be.
s	az egyes vermek nagysága byte-ban. Alapértelmezése 128. Ezt az értéket 32 és 512 közé állíthatjuk be.

Megjegyzés:

A DOS minden egyes hardver megszakítas esetén allokál magának egy vermet a fenti paraméterek alapján kialakított vermekből. A megszakítás lekezelése után a verem területe ismét felszabadul.

12. A DOS speciális lehetőségei

12.1 A standard I/O átirányítása

Valamennyi DOS parancs három perifériát használ:

- standard input;
- standard output;
- standard periféria a hibaüzenetek kiírására.

Az egyes parancsok - feladattól függően - további perifériákat használhatnak. A DOS lehetőséget biztosít a standard input és output átirányítására egy másik perifériára (pl. az output-ot a nyomtatóra, vagy egy lemezes file-ba). A hibaüzenetek mindig a képernyőn jelennek meg.

Az output átirányításához a DOS parancs után az alábbiak egyikét kell írni:

a) >*periféria*
b) > >*periféria*

A periféria vagy egy logikai periféria neve, vagy egy file leírása. Ebben az esetben

[*d:*][*út*]*file-név*[*.kit*]

alakúnak kell lennie. Az a) esetben, ha lemezes file-ról van szó, a DOS egy új file-t hoz létre. A b) esetben, ha már létezik a file, a DOS standard outputja hozzáíródik a megfelelő file tartalmához.

Megjegyzés:

1. Az a) esetben, ha már létező file-t adunk meg, akkor az output a file első rekordjába kerül tehát felülírja, törli a régi adatokat.
2. A b) esetben, ha a file még nem létezik, akkor a rendszer azt létrehozza.

Az input átirányítására az alábbit kell a DOS parancs után írni:

<*periféria* vagy

< [*d:*][*út*]*file-név*[*.kit*]

A program vagy parancs a megadott file-ról vagy perifériáról várja az inputot.

Megjegyzés:

1. Ha az átirányított standard input file-ban valamilyen hiba fordul elő (pl. helytelen paraméterezés), akkor a parancs befejezi működését, és a DOS prompt jelenik meg a képernyőn. (A standard input periféria ismét a billentyűzet.)
2. Standard input átirányítása esetén a teljes inputot a file-nak kell tartalmaznia. Amennyiben a program végigolvasta a standard inputot, ezután a DOS onnan már csak "file-vége" jelet olvas be. Amennyiben a programnak még valamire szüksége

van, soha nem kapja meg, és könnyen lemerevedhet. Ilyenkor csak a <Ctrl-Break>-kel szakítható meg.

Példa:

1. A következő parancs megadása esetén az aktuális lemezegység tartalomjegyzéke nyomtatóra kerül:

```
DIR >PRN
```

2. Ebben az esetben pedig a DIR outputja egy már létező file-hoz hozzáíródik:

```
DIR > >FIXLEMEZ.DIR
```

3. Ebben a példában a standard input átirányítását mutatjuk be. Tegyük fel, hogy a PROGRAM1.EXE a billentyűzetről olvasná be az adatait. Ezeket az adatokat mi már egy file-ban: (PROGRAM.ADA) elhelyeztük. (Ezt a sorszerkesztő (EDLIN) vagy a COPY parancs segítségével is megtehetjük.) Ezután a program indítása a következő módon is történhet:

```
PROGRAM1<PROGRAM.ADA
```

12.2 A standard I/O-k összekapcsolása

Ez a lehetőség biztosítja, hogy több program/parancs végrehajtását összekapcsoljuk úgy, hogy az egyik standard outputja a következő standard inputja legyen.

Ebben az esetben a DOS a főtartalomjegyzékben egy ideiglenes file-t alakít ki, aminek az azonosítója %PIPEx.$$$ lesz. A DOS gondoskodik arról, hogy az első parancs standard outputja a %PIPEx.$$$ file-ba kerüljön, s hogy a következő parancs onnan olvassa a standard inputot. Az esetleges hibaüzenetek a képernyőre kerülnek.

A standard I/O összekapcsolások lehetőségét "PIPE"-funkciónak vagy "PIPE" típusú parancsláncnak is nevezik.

Formája a következő: *parancs*[¦*parancs*]...

Az összekapcsolásra kerülő programok/parancsok azonosítóit a ¦ jellel kell elválasztani. Ha ez a jel a billentyűzeten nem található, akkor az <ALT> billentyű és az 1,2,4 numerikus billentyűk lenyomása (a klaviatúra jobb oldalán) pótolja.

Megjegyzés:

1. Az előző parancs/program outputja nem jelenik meg a képernyőn.

2. A %PIPEx.$$$ azonosítóban az x az ideiglenes file sorszáma.

3. A parancs1 ¦ parancs2 hatása azzal azonos, mintha egymás után kiadtuk volna a

```
parancs1 > %PIPEx$$$
parancs2 < %PIPEx.$$$
DEL %PIPEx.$$$
```

DOS parancsokat.

Példa:

1. A fejezet további részében ismertetésre kerül a SORT parancs. Ha a következő parancsot adjuk meg:

```
DIR ¦ SORT > RENDEZET.DIR
```

Ekkor a DIR outputja - ami az aktuális meghajtó tartalomjegyzéke - lesz a SORT inputja. Ezt a SORT rendezi, majd a SORT outputja a RENDEZET.DIR file-ba kerül. Később ez a TYPE paranccsal a képernyőre listázható.

12.3. Speciális DOS parancsok

12.3.1 MORE - lista oldalakra tagolás

A MORE ún. filter-parancs. Az adatokat a standard inputról olvassa és azt a standard outputra írja, de egy képernyőnyi adat átvitele után üzenetet küld, és szünetel a kijelzés. Ha lenyomunk egy tetszőleges billentyűt, akkor folytatódik az átvitel. Ez mindaddig ismétlődik, amíg valamennyi input adat átvitelre nem kerül, (illetve <Ctrl-Break>-kel meg nem szakítjuk a parancsot).

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

MORE < [*d:*][*út*\]*file-név*[*.kit*]

Megjegyzés:

Az angol, illetve a német nyelvű rendszerekben a következő üzenetet kapjuk:

- MORE -, illetve - FORTSETZUNG -

Példa:

A PROG1.ASM program sorait írjuk a képernyőre. Ha a képernyő megtelt, akkor üzenetet küld a rendszer, majd egy billentyű lenyomásával folytatódik a kijelzés. Egyben példát is látunk a standard input átirányítására.

A parancs a következő:

```
MORE < PROG1.ASM
```

Amennyiben a MORE parancsot a standard input átirányítása nélkül hívtuk meg (pl. elfelejtettük a '<' jelet hívni), a program a billentyűzetről vár inputot. Vagy adjunk <F6>-ot, majd <ENTER>-t - ekkor a program terminál, vagy abortáltassuk <Ctrl-Break>-kel.

12.3.2 FIND - keresés, válogatás

A FIND ún. filter-parancs. A parancs paraméterében meghatározott file (esetleg több file) rekordjaiban egy meghatározott karaktersorozatot keres. Azok a rekordok, amelyek a megadott karaktersorozatot tartalmazzák, a standard outputon kijelzésre kerülnek.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

```
FIND [/V][/C][/N]karaktersorozat[[d:][út]file-név[ kit ]...]
```

A paraméterek jelentése:

/V paraméter megadása esetén minden olyan rekordot kijelez, amelyik a megadott karaktersorozatot tartalmazza

/C paraméter megadása esetén csak azoknak a rekordoknak a számát írja ki, amelyek a megadott karaktersorozatot tartalmazzák

/N alkalmazása esetén a rekord relatív sorszámát, majd a rekordot is kijelzi egyezés esetén.

Megjegyzés:

1. A karaktersorozatot idézőjelek (") között kell megadni.
2. A globális file-név jelek (?,*) nem használhatók a file-ok kijelölésére.

Példa:

A FIND egy összetett alkalmazására mutatunk példát. A standard I/O összekapcsolásával egy lemez tartalomjegyzékében a .COM karaktersorozatot keressük.

```
A > DIR C¦FIND/V".COM"
```

A parancssorozat valamennyi .COM kiterjesztésű file-nevet kiírja.

12.3.3 SORT - rendezés

A SORT ún. filter-parancs. A standard inputról adatokat olvas be, azokat rendezi, majd a standard outputra írja.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

```
SORT [/R][/+n]
```

A paraméterek jelentése:

/R megadása esetén a rendezés alfabetikusan csökkenő lesz. Az alapértelmezés a növekvő sorrend. (Reverse - fordított: a B karakter az A előtt áll.)

/+n helyén egy nem negatív egész szám áll. Ez azt jelöli, hogy a rekord n-edik karakterétől (pozíciójától, oszlopától) történik a rendezés. Ha nem adunk meg értéket, akkor 1 az alapértelmezés, tehát a rekord első karakterétől történik a rendezés.

Példa:

A SORT egy összetett alkalmazására mutatunk példát. A standard I/O összekapcsolásával a tartalomjegyzék rekordjait a file-ok mérete szerint rendezzük és ezt nyomtatóra átirányítjuk:

```
DIR C: |SORT / + 14 > PRN.
```

Ha az outputot egy file-ba szeretnénk átirányítani, akkor ez a következő parancsokkal történhet:

```
DIR C: |SORT/ + 14 > RENDEZET.DIR
```

Megjegyzés:

1. A rendezendő file mérete legfeljebb 63 Kbyte lehet.
2. A tabulátor jeleket a SORT nem üres karakterként értelmezi.
3. A karaktereket a SORT bináris értékük szerint rendezi, kivéve a következő eseteket:

- a kisbetűket a-tól z-ig ugyanolyan súllyal veszi a rendezésnél figyelembe, mint a nagybetűket A-tól Z-ig, azok egyenlő eséllyel kerülnek - tehát előfordulási sorrendben - ugyanarra a helyre;
- a decimális 127-nél nagyobb ASCII értékű karaktereket a következő táblázat szerint veszi figyelembe:

ASCII érték (dec)	ASCII rendezési súly	ASCII érték (dec)	ASCII rendezési súly
128	67	153	79
129	85	154	85
130	69	155	36
131	65	156	36
132	65	157	36
133	65	158	36
134	65	159	36
135	67	160	65
136	69	161	73
137	69	162	79
138	69	163	85
139	73	164	78
140	73	165	78
141	73	166	166
142	65	167	167
143	65	168	63
144	69	169	169
145	65	170	170
146	65	171	171
147	79	172	172
148	79	173	33
149	79	174	34
150	85	175	34
151	85	225	83
152	89		

12.3.4 PROMPT - a prompt-karaktersor megváltoztatása

A parancs segítségével a képernyőn megjelenő prompt-karaktert, vagy karaktersorozatot (rendszerazonosítót) lehet beállítani.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

PROMPT [*szöveg*]

A "*szöveg*" lesz az új rendszer prompt szövegrésze. Ebben a szövegben előfordulnak speciális, két karakterből álló karaktersorozatok is. E karaktersorozat első karaktere a $. A második karakter a következő értékeket veheti fel:

$: a dollárjel
t: idő
d: dátum
p: aktuális tartalomjegyzék az aktuális meghajtón
v: a DOS verziója
n: az aktuális meghajtó betűjele
g: a > karakter
l: a < karakter
b: a ¦ karakter
q: az = karakter
h: a kurzor visszaléptetését és a megelőző karakter törlését jelöli
e: az ESCape karakter
_: a CR/LF (Carrige Return - Line Feed; kocsi vissza-soremelés) a képernyő következő sorának az elejére kerül a kurzor.

Egyéb értékek megadását a DOS figyelmen kívül hagyja. Ha a PROMPT parancsot "*szöveg*" nélkül adjuk meg, visszaáll az eredeti (azaz a "C>" alakú) DOS prompt.

Megjegyzés:

1. Ha a $ karakter maga is szerepel a "szöveg" karaktersorozatban, akkor azt $$ formában kell megadni.
2. A $e (ASCII-kód X"1B") megadásával escape-sorozatok állíthatók elő. Ezeket a kibővített képernyő- és billentyűzet-kezelés című alfejezetben ismertetjük.

Példa:

1. Példa olyan "szöveg"-re, amely szintén a normál DOS promptot állítja vissza:

PROMPT ng

2. PROMPT 123ABC hatására a rendszer promptja az 123ABC szöveg lesz.

3. Az új prompt karaktersorozat a rendszer dátum és a rendszer idő lesz, ha a következőt adjuk meg:

```
PROMPT $d$_$t
```

A dátum és az idő nem egy sorban kerül kijelzésre.

4. Ebben a példában a prompt karaktersorozat az aktuális meghajtó azonosítója lesz, de az escape-szekvencia megadásával a kijelzés "fordított" módon (reverse video) történik. A parancs a következő:

```
PROMPT $e[7m$n:$e[m
```

$e[7m: az inverz video kijelzés bekapcsolása,
$e[m: a normál kijelzés visszaállítása a képernyőn.

12.3.5 CTTY - alternatív konzol kijelölése

A parancs segítségével az aktuális input/output konzol egy alternatív konzolra változtatható, illetve az aktuális billentyűzetet és a képernyőt ismét konzolként használja a DOS.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

CTTY *egységazonosító*

A paraméter jelentése:

Az *egységazonosító* a következő lehet (a kettőspont használata nem kötelező):

AUX: az első aszinkron adapter,
COM1: az első aszinkron adapter,
COM2: a második aszinkron adapter,
COM3: a harmadik aszinkron adapter, (DOS 3.30-tól)
COM4: a negyedik aszinkron adapter, (DOS 3.30-tól)
CON: az aktuális billentyűzet, képernyő.

Megjegyzés:

1. A CTTY bármilyen karakteres perifériaazonosítót elfogad, így például a PRN-t, a nyomtatót is. Ez a rendszer leállásához vezet, mivel a nyomtató nem tudja végrehajtani a rendszer által kiadott input parancsokat (például: olvasási parancs).
2. A felhasználó saját karakteres perifériájának az azonosítóját is megadhatja, ha az input és output műveleteket is végre tud hajtani.

Példa:

1. A DOS az AUX egységet tekinti az aktuális konzolnak, ha a következő parancsot kiadjuk:

```
CTTY AUX
```

2. Az aktuális klaviatúra, képernyő eredeti hozzárendelés visszaállítása a következő paranccsal történik:

```
CTTY CON
```

12.3.6 EXE2BIN - átalakítás .EXE-ről .COM formátumra

A parancs lehetővé teszi, hogy a LINK programszerkesztővel készített .EXE kiterjesztésű programot átalakítsuk .COM formátumra. A .COM formátumú programot a parancsprocesszor parancsként hajtja végre.

Az átalakítás következtében a program mérete csökken és betöltése is gyorsabb lesz.

A .COM és az .EXE formátumról a 15. fejezetben (a PC DOS munkaterületei és vezérlő blokkjai) találhatók további információk.

Verzió: 2.11 - 3.30
A parancs típusa: Külső
A parancs formája:

EXE2BIN [*d:*][*út*]*file-név*[*.kit*] [*d:*][*út*][*file-név*[*.kit*]]

A paraméterek jelentése:

Az első paraméterrel az EXE formájú input programot határozzuk meg. A file-név kiterjesztésének az alapértelmezése: .EXE. A második paramétert csak akkor kell megadni, ha a COM formájú program neve változik vagy a nevének a kiterjesztése nem .COM lesz.

Megjegyzés:

1. Ha az első file nem .EXE formájú, akkor a következő üzenetet kapjuk:

   File cannot be converted, illetve
   Datei kann nicht umgestellt werden.

2. Az átalakítás nem végezhető el, ha az átalakítandó program 64 Kbyte-nál nagyobb méretű, illetőleg, ha a betöltési szegmenstől függő relokációs elemet tartalmaz (ilyen a STACK szegmens is!).
3. A .COM formátum az assembler nyelvű programok esetén az ORG assembler direktíva segítségével alakítható ki:

```
ORG 100H
START:
.
.
.
END START
```

12.3.7 SET - programkörnyezet kezelése

Minden végrehajtásra kerülő program előtt a memóriában egy lefoglalt tárterület helyezkedik el, amely tartalmazza a DOS aktuális paramétereit. Ez a terület definiálja a programkörnyezetet. A SET parancs lehetőséget ad a programkörnyezet lekérdezésére, átírására, tartalmának bővítésére. A programkörnyezet részletes ismertetése megtalálható a "PC-DOS munkaterületei és vezérlőblokkjai" című fejezetben.

Verzió: 2.11 - 3.30
A parancs típusa: Belső
A parancs formája:

SET [*kulcsszó* = [*paraméterérték*]]

A parancs megadható formái:

- A SET parancs önmagában áll, sem kulcsszó sem paraméterérték nem követi. Ilyen esetekben a képernyőre kiíródik a programkörnyezet valamennyi kulcsszava az aktuális paraméterértékekkel együtt. Pl.:

```
A>SET     ennek hatására kiíródik:
COMSPEC=C:\
PROMPT=$P$G
PATH=\;\SYSTEM
```

- A SET parancsot egy kulcsszó és egyenlőségjel követi: "SET prompt =". Ennek hatására a parancsban megadott kulcsszó és a hozzá tartozó paraméter törlődik a programkörnyezetből. Ezután a "SET" már csak az alábbi kiírást eredményezné a képernyőn:

```
COMSPEC=C:\
PATH=\;\SYSTEM
```

- A SET parancsban a kulcsszót követő egyenlőségjel után paramétert is megadunk. Ebben az esetben az így megadott kulcsszó és paraméter bekerül a programkörnyezetbe. Ha kulcsszó már szerepel benne, akkor a korábbi megadás törlődik. Például:

```
SET printscrn=yes
```

parancs hatására a programkörnyezetbe bekerül a "PRINTSCRN=yes" jelsorozat. Ha történetesen van olyan felhasználói program, amely abban az esetben, ha a PRINTSCRN értéke yes (vagy YES), minden üzenetet képernyőre és printerre is kiküld, különben csak képernyőre, úgy ez a program ezután printelni is fogja az üzeneteit.

Megjegyzés:

1. Amint az utolsó példában is látható, a PC-DOS a kulcsszót mindig nagybetűsre konvertálja, továbbá így hasonlítja és tárolja.
2. A PC-DOS csak az első példában is szereplő COMSPEC, PROMPT és PATH kulcsszavakat értelmezi. Ezek közül a COMSPEC a rendszer konfigurálásakor állítható. A másik kettő önálló parancsként is megadható. A kétféle megadás teljesen egyenértékű és azonos hatású.

3. A programkörnyezet mérete legfeljebb 32 Kbyte lehet.
4. Az érvényes programkörnyezet-szegmens címe a betöltött program PSP-jének (Program Segmens Prefix) hexadecimális 2C relatív címén található.
5. A rezidens programok környezetleírását már sem a SET, sem a PROMPT, sem a PATH parancs nem változtatja meg.
6. A környezeti változók a parancs-file-okban használhatók (lásd 7.1.2!).

12.3.8 NLSFUNC - ország-függő információk kezelése

Az NLSFUNC parancs segítségével lehetőségünk nyílik a kibővített ország-függő információk kezelésére. A program egy része a memóriában marad, annak segítségével látja majd el a DOS ezta funkciót.

Verzió: 3.30
A parancs típusa: külső
A parancs formája:

NLSFUNC [[*d:*][*út*]*file-név*.[*kit*]]

A paraméter jelentése:

Az NLSFUNC után szereplő file-név az ország-függő információkat tartalmazó file-t azonosítja. Ha a paramétert nem adjuk meg, akkor a CONFIG.SYS file-ban elhelyezett COUNTRY= parancsban szereplő file-t használja. Ha ott sincs megadva, akkor a COUNTRY.SYS file-t keresi.

12.3.9 CHCP - kódlap váltás valamennyi eszközre

A parancs segítségével megadhatjuk, hogy a DOS melyik kódlappal dolgozzon. A DOS valamennyi eszköze módosítja a kódlapot, feltéve, hogy ezt valamilyen formában (pl. egy MODE CP PREP paranccsal) megengedtük.

Verzió: 3.30
A parancs típusa: külső
A parancs formája:

CHCP [*kp*]

A paraméter jelentése:

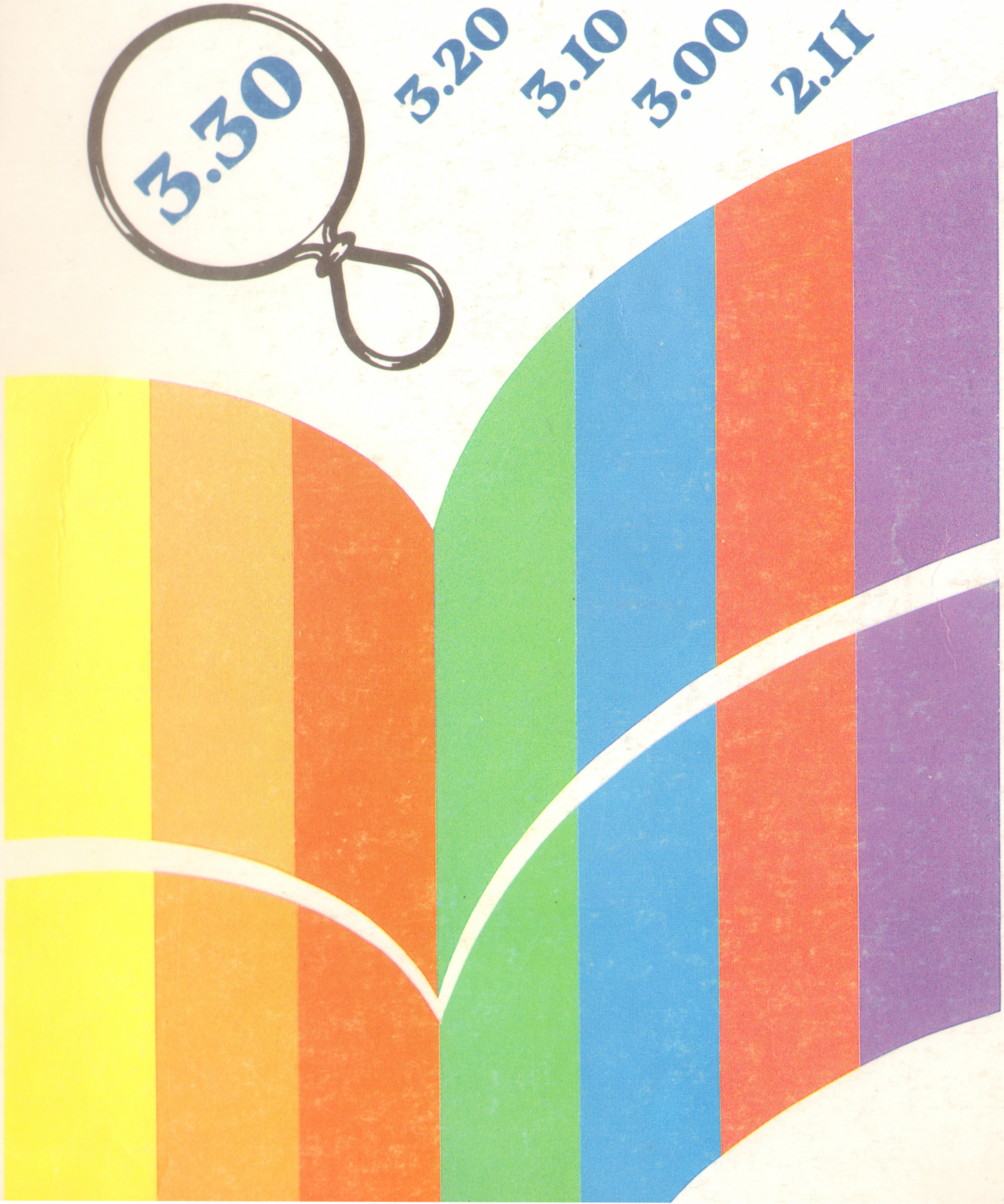
kp a kívánt kódlap sorszáma.

Megjegyzés:

A CHCP kiadása előtt le kell futtatni az NLSFUNC programot is.

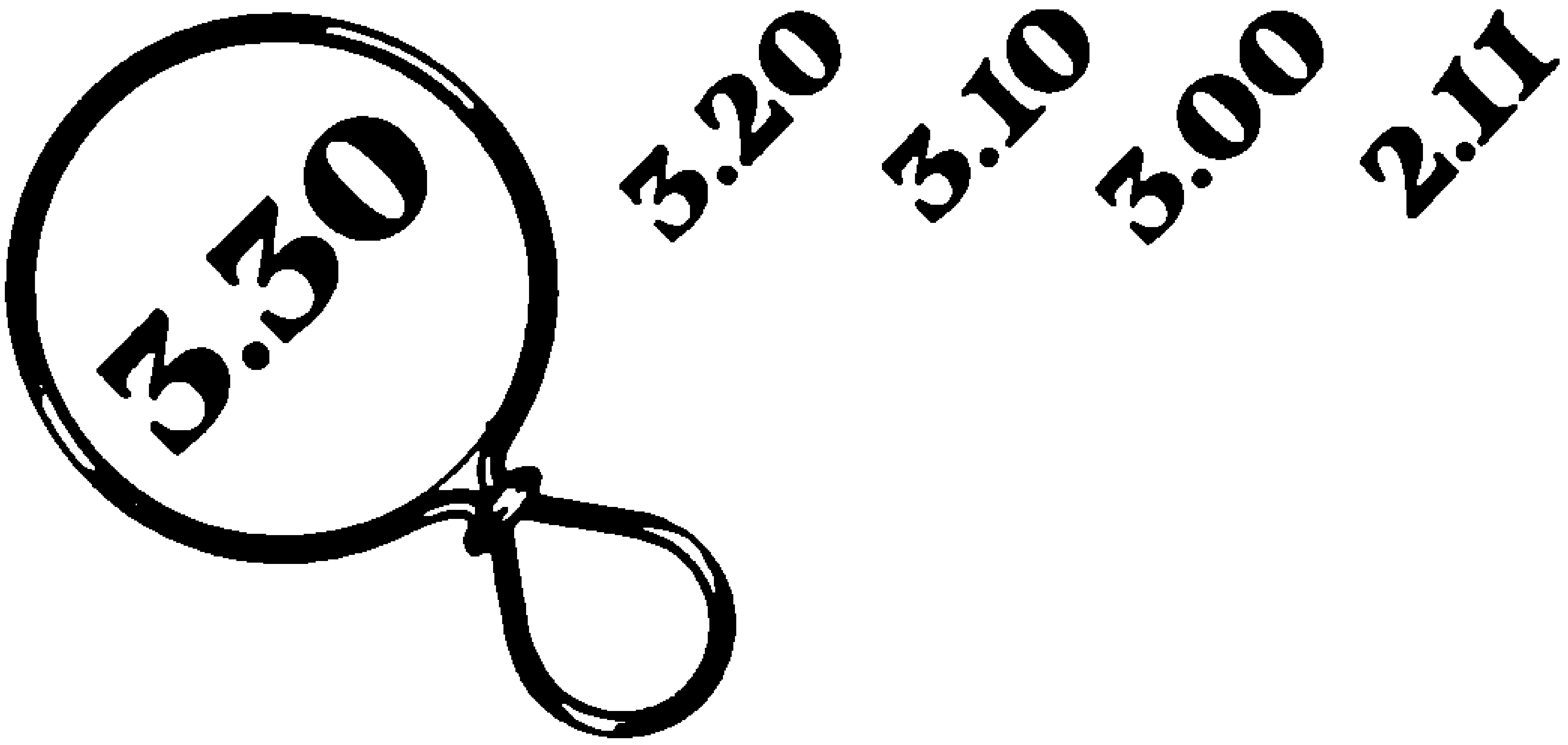
IBM PC DOS

II. A PC DOS FELÉPÍTÉSE


3.30
3.20
3.10
3.00
2.11

IBM PC DOS

II. A PC DOS FELÉPÍTÉSE


3.30
3.20
3.10
3.00
2.11

ALKALMAZÁSTECHNIKAI TANÁCSADÓ SZOLGÁLAT
BUDAPEST, 1990

Kónzó Zsolt

Szerkesztette: Szenes Katalin I-II.
Dr. Úry László III.

Készítette:	Bernáth István	1, 2, 8 és 9. fejezet
	Darvas László	3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16 és 17. fejezet
	Orosz Judit	10. fejezet
	Seres József	3, 4, 5, 6, 7, 11, 12, 13, 14, 15, 16, és 17. fejezet
	Úry László	3.xx kiegészítések

III kötet: Orosz Judit
Baki Zoltán
Pajor Gábor

Lektorálta: Éltető László
Szenes Katalin
Darvas László
Donát János

Munkatárs: Sziklai Klára

Kiadó: LSI Alkalmazástechnikai Tanácsadó Szolgálat
Felelős kiadó: dr. Kovács Magda
Témafelelős: Székely László
Szedés: LSI ATSz Info Osztály
Technikai szerkesztő: Nagy Olivér
Engedélyszám: 57152
ISBN II. kötet: 963 576 006/x
AGRO-PRINT Kft. Gyál 90-67/2
Felelős vezető: Tóth Antal

TARTALOMJEGYZÉK

I. kötet

Előszó 3
Tartalomjegyzék 5

1. FEJEZET

Bevezetés 11

2. FEJEZET

PC DOS alapismeretek 17

3. FEJEZET

DOS: file-kezelő parancsok 41

4. FEJEZET

DOS: lemezkezelési parancsok 60

5. FEJEZET

A fix lemez kezelése 81

6. FEJEZET

A rendszer működését befolyásoló parancsok és kapcsolók 90

7. FEJEZET

A parancs-file-ok 105

8. FEJEZET

A sorszerkesztő program (EDLIN) 115

9. FEJEZET

A nyomkövető-hibakereső program (DEBUG) 141

10. FEJEZET

Futtatható program előállítása - programszerkesztés 161

11. FEJEZET

A rendszer konfigurálása 171

12. FEJEZET

A PC DOS speciális lehetőségei 184

II. kötet

13. FEJEZET

A PC DOS belső struktúrája 202
13.1 A parancsfeldolgozó program (COMMAND.COM) 202
13.2 A PC DOS indítása 203
13.3 A PC DOS-nak a felhasználói programok által elérhető szolgáltatásai, függvényhívási és megszakítási rendszere 205
  13.3.1 A 8086-os processzor regiszterei 205
  13.3.2 A 8086-os processzor címképzési rendszere 206
  13.3.3 A megszakítás fogalma 207
  13.3.4 A PC DOS megszakításainak alkalmazása felhasználói programban 210
  13.3.5 A funkcióhívási rendszer 211
13.4 A DOS perifériakezelése 211
  13.4.1 Karakteres berendezések 211
  13.4.2 Blokkos berendezések 212

14. FEJEZET

Lemezkezelés és a felhasználói perifériakezelők szerkezete 213
14.1 A floppy és a winchester lemezek felépítése 213
  14.1.1 A boot szektor és a partíciós tábla 214
  14.1.2 A file leíró tábla (FAT) 217
  14.1.3 A tartalomjegyzék (directory) felépítése, altartalomjegyzékek 220
  14.1.4 Kiterjesztett DOS partíció felépítése 222
14.2 A felhasználói periferiakezelő program (handler, driver) 223
  14.2.1 A perifériakezelő program felépítése 223
  14.2.2 A perifériakezelő programokkal kapcsolatos tudnivalók összefoglalása és kiegészítése 225
  14.2.3 A parancsblokk és fejlécének felépítése 227
  14.2.4 A perifériakezelőt vezérlő parancsok részletes leírása 228
  14.2.5 Javaslat gyakorlati példára 234

15. FEJEZET

A DOS memóriakezelése, munkaterületei és vezérlőblokkjai 236
15.1 A DOS memóriatérképe 236
15.2 A programszegmens prefix (PSP) 237
  15.2.1 A PSP felépítése 238
  15.2.2 A PSP egy program indításakor 239
15.3 A programkörnyezet es a memórianyilvántartás 240
  15.3.1 Programkörnyezet 240
  15.3.2 A memórianyilvántartás 241
15.4 A file-kezelés és memóriaterületeinek szerkezete 243
  15.4.1 File-leíró blokk (FCB) 244
  15.4.2 Lemezátviteli terület (DTA) 246
  15.4.3 File-kezelés az FCB segítségével 246
  15.4.4 A DOS file-szám (handle) és használata 250

16. FEJEZET

DOS megszakítások ... 252

16.1 A 20-as megszakítás: program befejezése ... 252
16.2 A 21-es megszakítás: DOS funkció meghívása ... 252
16.3 A 22-es megszakítás: program kilépési cím ... 252
16.4 A 23-as megszakítás: Ctrl-Break kilépési cím ... 253
16.5 A 24-es megszakítás: kritikus hibák kezelése ... 253
16.6 A 25-ös megszakítás: közvetlen lemezolvasás ... 257
16.7 A 26-os megszakítás: fizikai szintű lemezírás ... 257
16.8 A 27-es megszakítás: befejezés, rezidens programok részére ... 258
16.9 A 2F-es megszakítás: multiplex megszakítás ... 258

17. FEJEZET

DOS funkciók ... 260

I/O műveletek karakteres berendezésekkel ... 260
Processzor-kezelő funkciók ... 260
Memória kezelés ... 260
Tartalomjegyzékkel kapcsolatos műveletek ... 260
FCB-s file-kezelő funkciók ... 261
File-számos file-kezelés ... 261
Egyéb rendszerfunkciók ... 261
File megosztás és hálózatkezelés ... 262

17.1 A DOS funkciók hibakezelése ... 263
17.2 A DOS funkcióhívások általános szabályai ... 264
17.3 A DOS funkciók ismertetése ... 266

Funkció ... **oldal**
"0" - program befejezése ... 266
"1" - tasztatúra input ... 267
"2" - kiírás képernyőre ... 267
"3" - soros vonali input ... 268
"4" - soros vonali output ... 268
"5" - karakter kiírása nyomtatóra ... 268
"6" - közvetlen konzol I/O ... 269
"7" - közvetlen konzol input képernyőre írással ... 269
"8" - konzol input képernyőre írás nélkül ... 270
"9" - karaktersorozat nyomtatása ... 270
"A" - pufferelt tasztatúra input ... 271
"B" - standard input egység állapotának ell. ... 271
"C" - a standard input puffer törlése, és egy standard input kezelő funkció hívása... 271
"D" - lemez visszaállítása ... 272
"E" - lemezegység kiválasztása ... 272
"F" - file megnyitása ... 273

"10" - file lezárása 273
"11" - adott FCB-beli file-név keresése tartalomjegyzékben 274
"12" - 11-es funkcióval megtalált file-t követő további file keresése 275
"13" - file törlése 275
"14" - file soros olvasása 276
"15" - file soros írása 276
"16" - file létrehozása 277
"17" - file átnevezése 277
"18" - a DOS számára FENNTARTOTT funkció 278
"19" - aktuális meghajtó kódjának visszanyerése 278
"1A" - a DTA címének beállítása 278
"1B" - aktuális meghajtó kódjának visszanyerése 278
"1C" - adott meghajtó FAT információina visszanyerése 279
"1D-20" - a DOS számára FENNTAROTT funkciók 279
"21" - random olvasás 279
"22" - random írás 280
"23" - file méret lekérdezése 280
"24" - a random rekord mező beállítása 281
"25" - a megszakítási (interrupt) vektor beállítása 281
"26" - új program szegmens előállítása 282
"27" - random blokk olvasása 282
"28" - random blokk írása 283
"29" - file azonosító keresése adott parancs-sorban 283
"2A" - dátum lekérdezése 285
"2B" - dátum beállítása 285
"2C" - rendszeridő lekérdezése 285
"2D" - rendszeridő beállítása 286
"2E" - VERIFY (lemezírás utáni ellenőrzés) kapcsoló beállítása 286
"2F" - lemez átviteli terület (DTA) címének lekérdezése 287
"30" - a DOS verziószám lekérdezése 287
"31" - programfutás befejezése úgy, hogy a program a memóriában marad 288
"32" - a DOS számára FENNTARTOTT funkció 288
"33" - a Ctrl-Break lekérdezése és beállítása 288
"34" - a DOS számára FENNTARTOTT funkció 289
"35" - megszakítási vektor lekérdezése 289
"36" - szabad lemezterület lekérdezése 289
"37" - a DOS számára FENNTARTOTT funkció 290
"38" - ország-specifikus információk lekérdezése 290
"39" - altartalomjegyzék létrehozása 291
"3A" - altartalomjegyzék törlése 292
"3B" - az aktuális tartalomjegyzék beállítása 292
"3C" - file létrehozása vagy meglévő file hosszának 0-ra állítása 293
"3D" - file megnyitása 293
"3E" - file lezárása 295
"3F" - olvasás file-ból, vagy egyéb perifériáról 295
"40" - írás file-ba vagy egyéb perifériára 296

"41" - file törlése adott tartalomjegyzékből 296
"42" - az írási/olvasási mutató átállítása 297
"43" - file attributumának lekérdezése vagy átállítása 298
"44" - periféria vagy file-leíró információ lekérdezése vagy átállítása 299
"45" - egy file számának duplikálása 302
"46" - file-szám duplikálás úgy, hogy a másodpéldányt a felhasználó adja meg 302
"47" - adott meghajtó aktuális tartalomjegyzékének lekérd 303
"48" - memóriafoglalás 303
"49" - lefoglalt memória felszabadítása 304
"4A" - lefoglalt memóriablokk méretének megváltoztatása 304
"4B" - program betöltése végrehajtatással vagy anélkül 305
"4C" - a program befejezése 306
"4D" - a hívott program befejezési kódjának lekérdezése 307
"4E" - elérési úttal megadott file keresése az első megfelelő file-ig 307
"4F" - további, a megadottnak megfelelő file-ok keresése, 4E hívás után 308
"50-53" - a DOS számára FENNTARTOTT funkciók 308
"54" - VERIFY kapcsoló értékének lekérdezése 308
"55" - a DOS számára FENNTARTOTT funkció 309
"56" - file átnevezése 309
"57" - file dátumának létrehozása és lekérdezése, időpontjának beállítása 310
"58" - memória helyfoglalás módjának lekérdezése illetve beállítása 310
"59" - a hiba leírásának lekérdezése 311
"5A" - file létrehozása egyedi névvel 312
"5B" - új file létrehozása 313
"5C" - file zárolása 314
"5E" - hálózati funkciók 314
"5F" - hálózati hozzárendelések beállítása, lekérdezése 315
"62" - programszegmens lekérdezése 316
"65" - kibővített ország-specifikus információk lekérdezése 317
"66" - globális kódlap beállítása/lekérdezése 318
"67" - periféria-kezelők maximális számának megadása 318
"68" - puffer kiürítése 318

FÜGGELÉKEK

A. A DOS üzenetei 319
  A.1 Perifériákkal kapcsolatos hibaüzenetek 319
  A.2 A DOS egyéb üzenetei 322
B. Parancsok, alparancsok csoportosított betűrendes listája 397
C. Parancsok, alparancsok formája 400
D. File-név kiterjesztések 404
E. Eszköznevek listája 404
Tárgymutató 405
Irodalomjegyzék 409

13. A PC DOS belső struktúrája

A PC DOS működésbe lépésétől kezdve a következő feladatokat hajtja végre:

- Betölti és működteti a perifériakezelő rendszerét.
- Feldolgozza és végrehajtja az aktuális konzolról érkező parancsokat.
- Kezeli és gazdálkodik a személyi számítógép perifériáival és központi memóriájával (azaz a gép erőforrásaival).
- Igény esetén betölti és elindítja a megfelelő felhasználói programokat.
- A felhasználói programok számára biztosítja a periféria kezeléséhez szükséges rutinrendszert..

Ezen feladatok elvégzését a DOS három, önálló file-ban található fő komponense az IBMBIO.SYS, IBMDOS.SYS és COMMAND.COM végzi. A IBMBIO.SYS és IBMDOS .SYS minden rendszerlemezen kötelezően az első két file-ként helyezkedik el. A rendszer lemezre írásakor ezek rejtett file-ok, ezért sem a DIR, sem a COPY, sem a DEL paranccsal nem kezelhetők.

A három főrészt kiegészíti a rendszerlemez 0. szektorában (a lemez felépítését lásd a "Lemezek felépítése" című fejezetben) található betöltő (boot) program. A DOS részét képezik továbbá a külső parancsok (ASSIGN, ATTRIB, BACKUP, COMP, DEBUG stb.) file-jai. A COMMAND.COM és a külső parancsok megléte a DIR parancs segítségével ellenőrizhető.

13.1 A parancsfeldolgozó program (COMMAND.COM)

A felhasználó a DOS rendszer ezen részével találkozik leggyakrabban. A rendszertöltés és inicializálás után ez a programrész kapja meg a vezérlést.

Feladatai:

1. Kapcsolattartás a felhasználóval:
    - Várakozási állapotnál bejelentkezés az aktuális prompt-tal.
    - Az operátori parancs beolvasása a konzolról, és ennek ellenőrzése.
    - A helyesen megadott parancsok végrehajtása esetén a programok indítása.
2. Perifériák működése közben fellépett hibák kezelése és üzenet küldése a konzolra (24-es megszakítás).
3. A DOS belső parancsainak (CHDIR, DIR, DEL stb.) végrehajtása.
4. <Ctrl-Break> billentyűk hatására keletkező megszakítás kezelése (23-as megszakítás).
5. Programbefejezés után visszatérés a DOS várakozási állapotba (22-es megszakítás).

A program négyféle módon léphet működésbe:

1. Várakozó állapotban a felhasználó a konzolról parancsot ad.

2. A gép bekapcsolása vagy <Ctrl-Alt-Del> egyidejű lenyomása után a DOS iniciáló része betölti a COMMAND.COM-ot és átadja számára a vezérlést.

3. A 2-es, 4-es, 5-ös feladat esetén a DOS törzse hívja meg.

4. A DOS ezen része úgy van szervezve, hogy újólag többször a tárba tölthető és indítható, ezért a felhasználói programok részére a DOS biztosítja a lehetőséget, hogy a 4B függvényhívással betöltsék és elindítsák. Ekkor a felhasználói programnak módjában áll a COMMAND.COM részére előre elkészített parancs átadására. Ilyen módon olyan felhasználói programok írhatók, melyek kezelik a DOS parancsokat is. Így működik például a BASICA-ban a SHELL utasítás.

Az utoljára betöltött parancsfeldolgozó programból az EXIT paranccsal tudunk visszalépni abba a programba, amely meghívta.

Az említett példánál a programok elhelyezkedése a memóriában a következő:

00000	Megszakítási tábla
	DOS állandó részei
	DOS adatterület, pufferek, felhasználói periféria kezelők
	DOS parancsfeldolgozó (COMMAND.COM rezidens része) 22H, 23H, 24H megszakítások kezelése
	DOS által betöltött rezidens programok
	Felhasználói verem .COM file-oknak (256 byte)
	szabad tárterület
	DOS parancsfeldolgozó (COMMAND.COM tranziens része), belső parancsok, parancs-file feldolgozó
A0000	rendszerterület, képernyő-pufferek
F6000	BASIC ROM területe
FE000	BIOS ROM

13.2 A PC DOS indítása

Minden felhasználóban felmerült már a kérdés, mi is történik a gép bekapcsolása, vagy a "meleg újraindítás" (warm reset: <Ctrl-Alt-Del>) és a DOS bejelentkezése közti néhány pillanatban. Nézzük meg most ezt a folyamatot.

Bekapcsolás esetén a gép a ROM BIOS FFFF:0 címén levő utasítástól indul. (Ez általában egy távoli ugrás a ROM BIOS önellenőrző rutinjába.) Ezután a BIOS program ellenőrzi a számítógép hardware elemeit és memóriáját. Ha ezzel végzett, először az A: egységben levő lemez legelső szektorát próbálja olvasni. Ha ez sikeres, a vezérlés rákerül a beolvasott kódra, s - amennyiben ez DOS lemez volt - ettől a pillanattól kezdve működik a PC DOS. Ha az A: meghajtóban nincs lemez, vagy az olvasás sikertelen volt, akkor - winchesteres gépnél - a winchester vezérlő ROM rutinja megpróbál olvasni az első (általában C: jelű) lemez aktív partíciójából. Ha ez sikeres, a vezérlés átadódik a beolvasott kódra, s az a fix lemez aktív partíciójáról próbál rendszert indítani. (Ha nincs aktív partíció, hibaüzenetet kapunk.) Ha a gépben nincs winchester lemez, ill. annak első szektora nem olvasható, a vezérlés a ROM BASIC-hez kerül.

Melegindítás (<Ctrl-Alt-Del>) esetén a hardware és memóriateszt elmarad, a ROM BIOS-nak (pl. winchester ROM-nak) a lemez legelső szektorát olvasó rutinja lép működésbe. Amennyiben gépünket programból szeretnénk újraindítani, bekapcsolási indítást a JMP FFFFH:0 , a melegindítást az INT 19H assembler utasítással válthatjuk ki.

Ezután a betöltő program ellenőrzi, hogy a lemezen megvan-e valamennyi rendszer-file. Hiány esetén hibaüzenetet küld a konzolra és leáll. Ha a rendszer-file-ok rendben voltak, akkor betölti az IBMBIO.COM file-t a memória aljára rezidens (állandóan ottmaradó) programként, és átadja neki a vezérlést.

Az IBMBIO.COM file-ban található programok hozzák a DOS-t tényleges indulási állapotba. Ezek a következő műveleteket végzik el:

- Betöltik a lehető legalacsonyabb címre az IBMDOS.COM rendszer-programot.
- Felépítik a DOS futásához szükséges belső táblákat.
- Inicializálják a DOS perifériakezelő rutinjait (handler-eket).
- Megnézik, hogy az aktuális lemezen van-e CONFIG.SYS file. Amennyiben igen, beolvassák, és annak alapján újra konfigurálják a rendszert. (A rendszer konfigurálásával kapcsolatos tudnivalókkal már a megfelelő fejezetben megismerkedhettek az olvasók.)
- Ezután kerül sor a COMMAND.COM betöltésére és elindítására.

A COMMAND.COM elindulása után, ha nem talál AUTOEXEC.BAT nevű file-t és nincs kijelölve beindítható program, akkor kiírja a verziószámot, bekéri a dátumot és időt (lásd a VER, a TIME és a DATE parancsokat). Ezt követőleg az aktuális prompt-tal bejelentkezik.

Ha van indításra kijelölt program, vagy AUTOEXEC.BAT file, akkor nem kerül sor verziószám, dátum, idő kiírásra, hanem a kijelölt programot indítja el a DOS.

Röviden áttekintve a fentieket, az indítás menete:

1. BIOS ellenőrző rutinok lefutása.

2. A BIOS beolvassa és elindítja a DOS betöltő rekordját.

3. A betöltő (boot) program beolvassa az IBMBIO.COM (vagy IO.SYS) file-t.

4. Az IBMBIO.COM betölti a rendszer további részeit, és azokat futáskész állapotba hozza.

5. CONFIG.SYS file létezése esetén az inicializáló program (IBMBIO.COM vagy IO.SYS) értelmezi azt, és annak megfelelően módosítja a rendszert.

6. A parancsfeldolgozó átveszi a vezérlést és az inicializálási ága lefut. (Bejelentkezés vagy kijelölt programok végrehajtása.)

A beindulás után a programok és főbb rendszerterületek tárbeli elhelyezkedését a "PC DOS memóriakezelése, munkaterületei és vezérlőblokkjai" (15. fejezet) című részben található memóriatérkép mutatja.

13.3 A PC DOS-nak a felhasználói programok által elérhető szolgáltatásai, függvényhívási- és megszakítási rendszere

Az előzőekből megismerhettük az operációs rendszer beindulását, a korábbi fejezetek pedig részletesen tárgyalták azt, hogy milyen közvetlen szolgáltatásai, parancsai vannak a PC DOS-nak. Magasszintű programnyelvek, célprogramok használata esetén ezek az ismeretek tökéletesen elégségesek a számítógép kezeléséhez. Igényes programok készítése során vagy software problémák megoldásakor viszont nagy segítség a DOS-nak a felhasználói programok számára nyújtott szolgáltatásainak ismerete. Ezek részletezése előtt bevezetjük a legszükségesebb gépi kódú programozási ismereteket.

13.3.1 A 8086-os processzor regiszterei

Az 1.2 fejezetben szó volt róla, hogy a PC DOS az Intel 8086 típusú és azzal kompatibilis mikroprocesszor alapú gépek operációs rendszere. Az ilyen processzorok a következő regiszterekkel rendelkeznek:

- 4 darab 16 bites általános regiszter: AX, BX, CX, DX. Ezek mindegyike 2 db nyolcbites regiszterként is elérhető. A felső nyolcbites rész jele H (high), az alsóé L (low), azaz például a MOV AX,0201H assembler utasítás egyenértékű a

```
MOV AH,02H
MOV AL,01H
```

utasításpárral.

- 4 db 16 bites szegmens regiszter: DS, ES, SS, CS. Ezen regiszterek a címképzésben vesznek részt. (DS = Data Segment, ES = Extra Segment, SS = Stack Segment, CS = Code Segment)
- 2 db 16 bites index regiszter: DI, SI. Az SI (Source Index) a string műveletekben a forrásterületre, a DI (Destination Index) pedig ilyenkor a célterületre mutat.
- 2 db 16 bites mutató regiszter: BP, SP. A BP (Base Pointer) regiszter segítségével a tár tetszőleges területén egyaránt futó ún. relatív vagy eltolható programok írhatók. (Relatív címképzés.) Az SP (Stack Pointer) a felhasználói veremtár eltolás értékét tartalmazza.
- Az IP regiszter mindig a következő végrehajtandó utasítás címére mutat.
- Az egy db 16 bites jelző (flag) regiszter felépítése:

Bit	Jelentés
15-12.	fejlesztésre fenntartva
11.	overflow - túlcsordulás
10.	direction - ciklusirány (növekvő vagy csökkenő)
9.	interrupt - megszakítás letiltva/engedélyezve
8.	trap - processzor lépésenkénti utasításvégrehajtás
7.	signum - előjel, ha 0, akkor az eredmény pozitív

6. zéró - eredmény nulla volt-e
5. fejlesztésre fenntartva
4. auxiliary carry - aritmetikai túlcsordulás jelzése
3. fejlesztésre fenntartva
2. parity - eredmény páros volt-e
1. fejlesztésre fenntartva
0. carry - túlcsordulás jelzése

A továbbfejlesztésre fenntartott bitek értéke mindig 0. A többi bit értékei a rájuk hatással lévő utasítások eredményeként változnak. Az, hogy melyik flag-et melyik utasítás állítja, a kézikönyv III. kötetében ismertetjük.

13.3.2 A 8086-os processzor címképzési rendszere

Ez a processzorcsalád valamennyi címet (utasítás-, verem-, adatcím) két részből képzi. A cím szegmense megmondja, hogy a gép memóriájának melyik 64 Kbyte-os területében vagyunk. A cím offset-je (eltolási értéke) mutatja azt, hogy a szegmens által megadott területen belül hányadik byte-ról van szó. A gyakorlatban ez úgy valósul meg, hogy a címzésben résztvevő szegmensregiszter értékének 16-szorosát a processzor hozzáadja a cím relatív helyét jelölő offset értékhez.

Mivel a 16-tal való szorzás a hexadecimális (16-os alapú) számrendszerben a szám végére egy zérus írását jelenti, ezért a címekkel való számolásnál célszerű a hexadecimális számok használata.

A hexadecimális számrendszerben 16 db számjegy van:

0-9-ig a megszokott számjegyek, továbbá
A = 10,
B = 11,
C = 12,
D = 13,
E = 14 és
F = 15

jelentéssel, azaz egy hexadecimális számban a 0-9, A, B, C, D, E, F jelek szerepelhetnek. A felismerhetőség miatt a nem igazi számjeggyel kezdődő ilyen számot szokás 0-val kezdeni.

Például BABA helyett 0BABA írása mutatja, hogy számról van szó. A hexadecimális számrendszer jelzése többféleképpen történhet. Szokásos jelölési formák:

X'BABA', 0BABAH (Macro Assembler),
&HBABA (BASIC),
$BABA (PASCAL).

(Ennek a számnak a decimális értéke:

11*4096 + 10*256 + 11*16 + 10 = 47802)

Mivel egy byte $2^8 = 256$ érték megkülönböztetésére ad lehetőséget, a 256 pedig éppen megegyezik 100H-val, ezért egy byte tartalma mindig megadható egy 2-jegyű hexadecimális számmal. (Pl.: 1111 1111B = 255 = 0FFH.) Hasonló megfontolással látható, hogy két byte tartalmának megadása hexadecimálisan 4-jegyű számok használatát igényli.

Mivel a továbbiakban mindig hexadecimális számokkal fogunk számolni, ezért, ha külön nem jelöljük, akkor mindig ilyet kell érteni, még akkor is, ha nem tartalmazza az A-F számjegyeket.

Első gyakorlatként nézzük meg a címképzést (hexadecimális számokkal). Legyen egy cím szegmens értéke 80, az offset értéke pedig 210. Ezt tömören így jelöljük: 80:210. (Ezt az írásformát használja a DEBUG program is, ezért célszerű ezt megszokni.) A fenti jelölés a

80 * 10 + 210 = 800 + 210 = 0A10

abszolút címet jelöli. Hasonló számolással látható, hogy

1000:0 = 10000 abszolút cím = decimálisan 65536 = 64 Kbyte

azaz a szegmensregiszterek segítségével 16*64 Kbyte, vagyis 1 Mbyte címezhető. Elvileg ez a 8086 alapú PC DOS által maximálisan kezelhető tártartomány.

Nem szóltunk még arról, hogy mikor melyik regiszter tartalmával számolja ki a processzor a címet. Általános szabály, hogy ha másként nem írjuk elő, akkor műveleti címképzésben mindig a CS, adatcímképzésben a DS, a verem címének a számolásában az SS, és sztring műveleti címeknél az ES és/vagy DS regiszter vesz részt.

Ha ettől el akarunk térni, akkor az utasítás elé írt úgynevezett regiszterprefix-szel (CS: vagy DS: vagy ES: vagy SS:) jelezhetjük, melyik regisztert akarjuk használni. Például:

```
MOV AX,[305]
```

utasítás betölti az AX regiszterbe a DS:305 és DS:306 memóriacímek tartalmát, de

```
CS:MOV AX,[305]
```

a CS:305, CS:306 memóriacímek tartalmát tölti AX-be.

13.3.3 A megszakítás (interrupt) fogalma

A DOS megszakítási rendszerének megértéséhez példákat is fogunk adni. E példákhoz elég lesz a MOV utasítás ismerete, ennek hatása a gépen is jól követhető.

Azok érdekében, akik még nem programoztak 8086 assemblerben, teszünk egy kitérőt, és ismertetjük a MOV-ot, amely már a fenti példában is szerepelt.

A MOV utasítás 8 és 16 bites adatok mozgatására használható, és az utasításkódon kívül két operandust tartalmaz. Az adatmozgatás történhet regiszterek, regiszter és memória, memória és regiszter, valamint két memóriacím között. A második operandus, amely mindig a forrást jelöli, lehet konstans is. Az operandusok a következő megkötésektől eltekintve szabadon alkalmazhatók:

- a két operandusnak egyforma hosszúnak kell lennie (nem mozgathatunk 16 bites regiszterbe 8 bites regisztert),
- nem lehet mindkét operandus szegmensregiszter,
- a cél operandus nem lehet a CS regiszter,
- minden olyan esetben, amikor valamelyik operandus memóriatartalom, és nem egyértelmű, hogy 8 vagy 16 bites műveletről van-e szó, az operandusok előtt a BYTE PTR vagy a WORD PTR prefix-szel jelezni kell a művelet típusát.

Példa:

```
MOV DS,AX                 ; 16 bites, DS-be betölti AX tartalmát
MOV AL,CH                 ;  8 bites, AL-be betölti CH tartalmát
MOV [30A],DX              ; 16 bites DS:30A,30B címekre tárolja DX tartalmát
MOV [30A],BL              ;  8 bites DS:30A címre tárolja BL tartalmát
MOV CX,1000               ; 16 bites CX-be ír 1000-t
MOV BYTE PTR [230],0      ;  8 bites, a DS:230-at kinullázza.
```

Példák hibás MOV utasításokra:

```
MOV  [230],[400]          ; nem megengedett operanduspár
MOV  [230],100            ; nem megállapítható a művelet típusa
MOV  100,AX               ; konstans érték nem lehet az első operandus
MOV  DS,CS                ; szegmensregiszterek között a közvetlen adatmozgatás
                          ; nem megengedett
MOV  AX,CH                ; operandusok hossza eltér
```

Nem foglalkozunk a logikai, aritmetikai és sztringmozgató műveletekkel. A vezérlést irányító utasítások (feltétlen és feltételes ugrások, közeli és távoli szubrutinhívások, software interrupt hívás) közül csak a software interrupt hívás ismerete szükséges a PC DOS funkcióhívási és megszakítási rendszerének megértéséhez. Ezért a továbbiakban ezt ismertetjük.

Először ismerkedjünk meg az interrupt (megszakítás) fogalmával. A számítógép működéséhez mindenképpen szükséges, hogy bizonyos feltételek fellépése esetén félbeszakítson - megszakítson - minden műveletet, és a félbeszakítást kiváltó esemény szerint cselekedjen, majd folytassa az eredeti programot. A programnak azt a helyét, ahol félbeszakadt, megszakítási pontnak hívjuk. Jól érzékelhető a megszakítás a billentyűzet esetében. Tudjuk, hogy egy program futása a <Ctrl-Break> billentyűvel általában félbeszakítható. Ez azért lehetséges, mert minden billentyű leütésekor megszakítás lép fel. Az aktuális program futása felfüggesztődik, működésbe lép a billentyűmegszakítást kezelő rutin, amely kiolvassa a leütött billentyű pozícióját, és a módosító billentyűk helyzete szerint kialakítja a megfelelő kódot, amit a billentyűzet-pufferben tárol. Ha nem érzékelt speciális kezelést igénylő billentyűkombinációt, akkor visszatér a megszakítási pontra, különben a kijelölt feladat szerint cselekszik, például visszatér a DOS-hoz, vagy rendszert tölt, vagy képernyőt nyomtat és csak azután folytatja a programot.

Nézzük most végig lépésekre bontva egy megszakítás lefolyását:

1. Fellép egy megszakítási igény.

2. A processzor megvizsgálja, hogy a megszakítás engedélyezett-e. (Ez programból is befolyásolható, mégpedig a STI = megszakítás engedélyezése és CLI = megszakítás tiltása utasításokkal.)

3. Engedélyezett megszakítás esetén az alábbi módon meghívásra kerül a megszakítás sorszáma szerinti megszakítás rutin. A memória 0:0 címétől kezdődően egy címtáblázat található. A táblázat minden eleme 4 byte hosszú, és egy-egy cím offsetjét és szegmensét tartalmazza. A processzor ezen táblából veszi a meghívandó rutin címét. Mielőtt ennek a rutinnak átadná a vezérlést, a veremtárba elrakja az állapotregisztert, a következő utasítás szegmensét, és a címét. Ez a vezérlésátadási folyamat software úton is előidézhető az INTn utasítással. Az utasításban az n tetszőleges (FF-nél kisebb, vagy egyenlő) szám lehet. A processzor ilyen számú megszakítás-hívást hajt végre. Például az INT 21 hatására a megszakítás vektortábla 21. eleméről (0:84-87) veszi az ugrási címet.

4. A megszakítás-kezelő rutin, miután elvégezte a feladatát, kiadja az IRET utasítást. Ennek hatására a vezérlés visszakerül a veremtárban található szegmens és offset által meghatározott címre úgy, hogy közben mind az állapotregiszter, mind a veremmutató (stack pointer) visszaáll eredeti értékére.

Az interrupt vektortábla elemei közül a 0-0F (a 3-as és az 5-ös kivételével) megszakítások hardware feltételek kezelését végzik. A továbbiak az alsószintű perifériakezelő rendszer, a BIOS egyes rutinjainak a meghívására, illetőleg az ezekhez szükséges paraméterek tárolására szolgálnak.

Az egyes megszakításokat külön fejezetben tárgyaljuk, a DOS funkciókkal együtt.

A fent elmondottakat IBM PC számítógépünkön a következőképpen nézhetjük meg:

```
  A>DEBUG                   elindítjuk a debug programot
  -d0:0040,44               parancs a megszakítási vektor 10. elemeinek kiírására
  000:0040  65 F0 00 F0     (a DEBUG válasza)
  -a100                     parancs: cs:100-tól utasítás beírása
  1059:100 int 10           (az int 10 utasítást írtuk be)
  1059:102 int 3            (az int 3 utasítást írtuk be)
  -r                        (regiszterek kiíratása)

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1057 ES=1057 SS=1057 CS=1057 IP=0100 NV UP EI PL NZ NA PO NC
1057:0100 CD10          INT 10

  -t                        (az INT 10 utasítás indítása - trace)

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFE8 BP=0000 SI=0000 DI=0000
DS=1057 ES=1057 SS=1057 CS=F000 IP=F065 NV UP DI PL NZ NA PO NC
F000:F065 E99D45        JMP 3605
```

ez a következő utasítás; látható, hogy címe megegyezik a kiíratott vektortábla elem tartalmával

```
  -q                        kilépés a DEBUG programból
```

13.3.4 A DOS megszakításainak alkalmazása felhasználói programban

A DOS a saját feladatainak elvégzéséhez a 20-3F megszakításokat használja. Ezek közül

- a program befejezését (20, 22, 27),
- a <Ctrl-Break> kezelését (23),
- a funkcióhívások alatt fellépő hibák kezelését (24),
- a fizikai lemezkezelést (25, 26) és
- a rendszer (funkció) hívást (21) végzők

egyértelműen definiáltak és leírásuk alapján szabadon használhatók. Ez azt jelenti, hogy a felhasználó a leírásban megadott paramétereket beállítja a megfelelő regiszterekben, és kiadja a megfelelő megszakítás hívást. A hívást követő utasításokkal kell a hívás után a regiszterek tartalmát feldolgozni, ellenőrizve a hibátlan lefolyást, illetőleg megállapítani az esetleges hiba okát.

Konkrét példán nézve:

Az A: lemezegységben lévő lemez második logikai szektorának beolvasása a DS:1000 címre a következő assembler utasításokkal lehetséges:

```
MOV AL,0        ; A egység beállítása
MOV CX,1        ; 1 szektor olvasása
MOV DX,1        ; első szektor
MOV BX,100      ; pufferterület első címe
INT 25          ; olvasás lemezről (fizikai)
.
.
.
```

Az INT 25 befejeződése után a ds:1000 címen meg fog jelenni az A egységben lévő lemez 0. oldalán lévő második szektor tartalma. (Az abszolut szektor sorszámozás zéróval indul - lásd a lemezek felépítésével foglalkozó fejezetet.)

Az egyes megszakítások leírása és részletes ismertetése a DOS általános struktúráját leíró fejezetek után, a 16. fejezetben található.

Végezetül felhívjuk a figyelmet arra, hogy a 22 és a 23 megszakítások kezelése eltér, ugyanis itt a felhasználónak kell igény esetén a megfelelő címeket az interrupt vektortáblában elhelyeznie. Az így elhelyezett címekre a megfelelő feltétel teljesülésekor a DOS adja át a vezérlést.

Hasonlóan nem hívható a felhasználó által a 24-es interrupt sem. Ez úgy használható, hogy a felhasználó a leírások alapján elkészíti a saját hibakezelő rutinját, és ennek a rutinnak a címét helyezi el az interrupt vektor megfelelő elemében. Így hiba esetén a DOS a felhasználó rutinját fogja meghívni. (Az interrupt vektor egyes elemeinek kicserélése a 25 és 35 alfunkcióhívásokkal oldható meg a legegyszerűbben.)

13.3.5 A funkcióhívási rendszer

Az előző alfejezetből látható, hogy a DOS perifériák kezeléséből és a felhasználóval, a felhasználói programmal való kapcsolattartásból csak néhány igen alapvető dolgot old meg önálló megszakítással. A legtöbb feladatot az egyetlen kiemelt megszakítás (a 21-es) megfelelő paraméterezésével végezhetjük el. Ez a megszakítás a DOS funkcióhívás, másnéven a rendszerhívás. Ez igen sokrétű szolgáltatást biztosít, és sok feladat (pl. kiírás képernyőre, file-kezelés, programbefejezés) többféle módon is megoldható. Ennek elsősorban az az oka, hogy az újabb javított, bővített verziók a kompatibilitás miatt változatlanul megtartják a korábbi hívásokat, ugyanakkor a bővítés sokszor csak a korábbi feladatok újabb módon való megvalósításával volt lehetséges.

Felmerült például az igény, hogy a program lefutása után meg lehessen állapítani a sikeres vagy sikertelen működést. Erre vezették be a 2.0 verziótól kezdődően a 4C eljárást, a program befejezése funkcióhívást. Ez használható a 0 funkcióhívás helyett, de több lehetőséget nyújt annál.

A funkcióhívások közös jellemzője, hogy a hívás sorszámát mindig az AH regiszternek kell tartalmaznia. A többi paramétert a megfelelő regiszterekben kell megadni. Az AH és a többi regiszterek beállítása után az INT 21 kiadásával lehet a megfelelő funkcióhívást aktivizálni.

Például nézzük meg a DOS verziószám lekérdezését, azaz azt, hogy a program milyen verziójú DOS alatt fut. Erre a 30-as funkcióhívás szolgál, amelynek nincs bemenő paramétere. Az aktuális verziószámot az AX-ben kapjuk vissza. A szükséges utasítások:

```
MOV AH,30
INT 21
```

a visszatéréskor az AX-ben megkapjuk a verziószámot.

13.4 A DOS perifériakezelése

Az előző alfejezetben bemutattuk, hogy milyen módon szervezi az operációs rendszer a munkáját, és hogy ebbe hogyan kapcsolódhat be a felhasználó. Most rátérünk a perifériák (eszközök, berendezések) kezelésére. A DOS kétféle módon, karakteresen és blokkosan kezel perifériát.

13.4.1 Karakteres berendezések

Ezeknél az eszközöknél a DOS az írás-olvasási műveleteket karakterenként végzi. Az egyes berendezések megkülönböztetése egy maximum 8 karakter hosszú névvel történik. A rendszer karakteresen kezelt perifériáinak nevei:

CON - konzol I/O egység
NUL - „üres" periféria
AUX - soros vonali I/O egység

COM*i*	-	soros vonali I/O egység. *i* = 1, 2, 3 vagy 4. A 3 illetve 4 érték csak a DOS 3.30 verziótól kezdve használható.
CLOCK$	-	rendszeróra
LPT*i*	-	párhuzamos vonali kimenet. *i* = 1, 2 vagy 3.

Az első kettő kivételével a karakteres berendezések használhatósága a gép kiépítettségén is múlik.

Ezeken kívül a felhasználó maga is definiálhat ilyen perifériákat. Ekkor egyrészt el kell készítenie a megfelelő perifériakezelő rutinrendszert megfelelő formában, másrészt a készülék nevét, a kezelőrutint tartalmazó file nevét és a rutin belépési pontjait közölnie kell a DOS-szal. (A CONFIG.SYS file DEVICE=... parancsában, illetőleg a perifériakezelő rutint tartalmazó file elején a DOS által megkövetelt formában.)

Mivel saját perifériakezelő rutin definiálásának bonyolult szabályai vannak, ezért azokkal külön rész foglalkozik.

13.4.2 Blokkos berendezések

A rendszer a lemezegységeket (hajlékony és fix lemezegységek) blokkos átvitellel kezeli, azaz tényleges fizikai adatátvitelt csak meghatározott (pl. 1 Kbyte) adatmennyiséggel hajt végre. A felhasználó írási/olvasási igényeit a DOS pufferezés segítségével elégíti ki. Ez azt jelenti, hogy ha egy lemez-file-ba karakterenként visszük is ki az adatokat, akkor az ténylegesen csak az 1024. byte kiírása után kerül fel a lemezre. Az utolsó töredék blokk kiírása mindig a file lezárásakor történik.

A blokkos perifériákat a DOS sorszámmal azonosítja, azonban a könnyebb kezelhetőség miatt az operátori parancsokban (DIR, DEL, REN stb.) a sorszámokat az ABC megfelelő betűje (A = 0, B = 1, ...) helyettesíti.

Saját perifériakezelőt blokkos berendezésekre is lehet készíteni. Ezekre is érvényesek a karakteres berendezéseknél fent felsorolt szabályok.

14. Lemezkezelés és a felhasználói perifériakezelők szerkezete

Ez a fejezet a PC DOS operációs rendszer perifériakezelését ismerteti.

Első részében a leggyakoribb külső adattárolók, a cserélhető lemez-(floppy) egység és a fix, vagy másként winchester lemezegység (Winchester disk/fix disk drive) logikai felépítését, táblázatait találhatja az Olvasó.

A fejezet második része a felhasználó által készíthető perifériakezelő rutinokkal kapcsolatos ismereteket foglalja össze.

14.1 A hajlékony és a fix lemezek felépítése

A DOS operációs rendszer igen nagy előnye, hogy az MS DOS és PC DOS rendszer alatt előállított lemezek szinte géptípustól függetlenül olvashatók, írhatók. Ez annak köszönhető, hogy a rendszer szerves részét képezi a lemezek egységes formátuma.

Mielőtt ezt ismertetnénk, először nézzük meg a lemez fizikai felépítését:

A lemezen az információ körkörösen, úgynevezett sávokon (track) helyezkedik el. A koncentrikus köröket alkotó sávokon belül floppy-nál lemezformátumtól függően 8 vagy 9 darab 512 byte-os szektor található (winchesternél több szektor helyezkedik el a sávokon). Ez a lemezen az információtárolás fizikai alapegysége.

A lemez lehet egy- vagy kétoldalas; szimpla, dupla, illetve négyszeres sűrűségű és nagy kapacitású.

Az egyoldalas lemeznek csak az alsó felén van információ. A szimpla sűrűségű lemezen 20, a duplán 40, a négyszeres sűrűségűn és a nagy kapacitásún 80 sáv található. (Megjegyzés: a forgalomban lévő gépek nem használnak szimpla sűrűségű lemezt.)

A winchester lemezben több, a cserélhető lemezhez hasonló, egymáshoz erősített mágneses lemez található. Ezeken mind a lemezoldalak, mind az egy oldalon található sávok, mind pedig a sávokon belüli szektorok száma több lehet, mint a cserélhető lemez megfelelő értékei.

Ahhoz, hogy a lemezen egy adatot megkeressünk, három jellemzőt kell megadni:

- az adatot tartalmazó sávot,
- az oldalt és
- a szektort.

A DOS az egységes kezelést úgy valósítja meg, hogy eltekint a szektorok tényleges elhelyezkedésétől, úgy veszi, mintha ezek folytonosan helyezkednének el. Így képez egy 0-tól induló logikai sorszámot az adatterületek (szektorok) azonosítására.

A logikai sorszámot a következőképpen kell elképzelni:

A 0. oldalon található 0. sáv első szektora az 1-es, második szektora a kettes, majd ez így folytatódik a sáv utolsó szektoráig. A következő sorszámot egyoldalas lemez esetén az 1-es sáv 1-es szektora, míg kétoldalas lemez esetén az 1-es oldal 0. sáv 1-es szektor kapja. Kettőnél több oldalas (több fejes) Winchester lemezeknél az 1-es oldal 0-ás sávja következik. Ilyen rendszer szerint folytatódik a szektorok sorszámozása. Az utolsó sorszám az 1-es - illetőleg winchesternél a legmagasabb sorszámú - oldal utolsó sávjának az utolsó szektorát jelzi.

Ezek után tekintsük át a hajlékony (floppy) lemezek felépítését:

0. szektor: **DOS boot szektor.** Az itt elhelyezkedő töltőprogram arra szolgál, hogy rendszertöltés esetén betöltse a lemezről a DOS további részeit és elindítsa azokat,
1. szektor: Innen kezdve k szektoron keresztül található az ún. **file leíró tábla (FAT** = File Allocation Table)
1 + k. szektor: Innen kezdve a file leíró tábla másolata található.
1 + 2k. szektor: Ettől kezdődően helyezkedik el a főtartalomjegyzék.
1 + 2k + n. szektor: Ettől kezdve találhatók a DOS és a felhasználó file-jai.

A winchester-ek felépítése kicsit összetettebb, ugyanis nagyobb kapacitása miatt biztosították a lemez (legfeljebb négy) részre oszthatóságát. Egy-egy összetartozó rész neve partíció.

A partíciók számát és méretét a felhasználó állíthatja be az FDISK parancs segítségével. (Ennek részletes leírása az 5.1 fejezetben található.)

A több partícióra való oszthatóság lehetősége elsősorban azt a célt szolgálja, hogy a winchester-rel ellátott gépet többféle operációs rendszerrel is lehessen használni. Ekkor az egyes operációs rendszerek egy-egy partíciót foglalnak el, mégpedig fizikailag összefüggő területen.

A partíciók kezelése egy úgynevezett partíciós tábla segítségével történik. A partíciós tábla a winchester fizikailag legelső szektorán helyezkedik el, és tartalmazza a partíciók azonosításának és elérésének az adatait.

A DOS partíció szerkezete megegyezik a cserélhető lemezek szerkezetével, azaz a tényleges kezdetétől függetlenül (ez csak a partíciós táblából állapítható meg) a 0. logikai DOS szektorral kezdődik, amely a DOS boot szektort tartalmazza. Ezt követi a file allokációs tábla és ennek másolata.

A rendszerterületek közül az utolsó a főtartalomjegyzék. Az egyes területek kezdete, mérete és típusa a boot szektorban található.

A boot szektor és a partíciós tábla felépítését a következő szakasz ismerteti részletesen.

14.1.1 A boot szektor és a partíciós tábla

A boot szektor, amely a cserélhető lemezen a 0. oldal 0. sáv első szektora, illetőleg a fix lemez esetén a DOS partíció 0. logikai szektora tartalmazza a PC-DOS rendszer betöltését elindító programot. Tartalmazza továbbá a program paramétereiként, meghatározott helyen, mindazokat az adatokat, amelyek a lemezen található DOS táblázatok feldolgozásához szükségesek. **Ennek felépítése a következő:**

Adatmező helye byte-ban	Adatmező tartalma
00-02.	Egy ugró utasítás a BOOT program további részére. Ez vagy 3 byte hosszú közeli (near) ugrás (E9H) vagy 2 byte-os relatív (short) ugrás (EBH), melyet egy NOP követ.
03-0B.	Egy 8 byte hosszú azonosító, mely a DOS nevéből és verziószámából áll.
0B-0C.	* Szó, amely egy szektor méretét tartalmazza byte-okban.
0D.	* Az egy logikai blokkban lévő szektorok számát tartalmazó byte. Megmutatja, hogy a FAT egy egysége hány szektort foglal magába. Ha tartalma például 4, akkor a lemezen lefoglalt legkisebb terület 512 byte-os szektorméret mellett 4*512 = 2048 byte = 2 Kbyte. (A DOS a tartalomjegyzékben látható file-méretet 2 Kbyte többszörösére fogja kerekíteni és ekkora területet használ el az adathordozóról.)
0E-0F.	* Szó, amely a foglalt szektorok számát tartalmazza. Ezek a szektorok a 0., a boot rekord kezdetétől helyezkednek el, és megelőzik a FAT első kópiáját.
10.	* A lemezen található FAT másolatok darabszámát (ez általában 2) tartalmazó byte.
11-12.	* Szó, amely megadja, hogy a főtartalomjegyzékben maximum hány bejegyzés lehet.
13-14.	* Szó, amely az adathordozón elhelyezkedő szektorok darabszámát tartalmazza. (Az adathordozó valamennyi szektora, beleértve a boot szektort, a FAT-ot és a főtartalomjegyzéket tartalmazó szektorokat stb.)
15.	* Az adathordozó típusát jelző byte. Szerkezetét és az érvényes típusjelzők listáját a hozzá tartozó formátumokkal együtt később ismertetjük.
16-17.	* A FAT egy másolatának mérete szektorokban.
18-19.	Szó, amely az egy sávon található szektorok számát tartalmazza.
1A-1B.	Szó, amely a lemezoldalak számát tartalmazza.
1C-1F.	Dupla szó, amely a rejtett szektorok számát tartalmazza. Ezek a szektorok a DOS boot rekord előtt helyezkednek el, azaz cserélhető lemezen itt nulla áll, winchester-nél pedig a DOS partíciót megelőző szektorok száma.
20-23.	Dupla szó. Jelenleg a DOS nem használja. Célszerű felhasználása: ha a 13-14. szó 0, akkor ez a duplaszó jelzi a logikai szektorok számát. Ez 32 Mbyte-os.
24.	Az indító (boot) program, melynek tényleges kezdete az első két vagy három byte-on található ugró utasításból állapítható meg.

A csillaggal megjelölt részek együtt alkotják az ún. BIOS paraméter blokkot (BPB). (A kiterjesztett DOS partíció boot szektorának leírása a 14.1.4 részben található.)

A partíciós tábla a Winchester lemez 0. sávjának 0. szektorában található. A szektor elején egy boot program helyezkedik el. Ez akkor kapja meg a vezérlést a ROM BIOS-tól, ha az A: lemezegység üres. A program feladata a rendszertöltésre kijelölt partíció kiválasztása és a kiválasztott partíció boot rekordjának aktivizálása. A partíció boot rekordja a továbbiakban ugyanúgy működik, mint a cserélhető lemezeken található boot program. (A rendszertöltés menete megtalálható "A PC-DOS belső struktúrája" című 13. fejezetben.) A boot rekordot követően az 1BE címtől kezdődően található a partíciós tábla, amelynek felépítése a következő:

Mező címe byte-ban	**Mező tartalma**
1BE	Az első rendszer töltést jelző byte-ja. Ennek tartalma alapján keresi ki a boot rekord a betölthető partíciót. Lehetséges értékei: • betölthető partíció esetén - hexadecimális 80, • nem boot-olható partícióknál - 00 (A négy közül csak egy partíció tartalmazhat 80-at.)
1BF-1C1	Az első partíció kezdetének címe a lemezen. Első byte tartalma: lemezoldal Második byte tartalma: szektor Harmadik byte tartalma: sáv
1C2	Rendszer jelző byte: • 00 ismeretlen • 01 elsődleges DOS 12 bites FAT • 04 elsődleges DOS 16 bites FAT • 05 kibővített DOS
1C3-1C5	Az első partíció végének címe. Első byte tartalma: lemezoldal Második byte tartalma: szektor Harmadik byte tartalma: sáv
1C6-1C9	Duplaszó, mely a partíció kezdetének logikai szektorcímét tartalmazza.
1CA-1CD	Duplaszó, mely a partíció méretét tartalmazza.
1CE	Második partíció rendszertöltést jelző byte-ja (tartalmát lásd az 1. partíciónál).
1CF-1D1	A második partíció kezdetének címe a lemezen.
1D2	A második partíció rendszer jelző byte.
1D3-1D5	A második partíció végének címe.
1D6-1D9	Duplaszó, mely a második partíció kezdetének logikai szektorcímét tartalmazza.
1DA-1DE	Duplaszó, a második partíció méretét tartalmazza.

1DE	A harmadik partíció rendszertöltést jelző byte-ja.
1DF-1E1	A harmadik partíció kezdetének címe a lemezen.
1E2	A harmadik partíció rendszer jelző byte.
1E3-1E5	A harmadik partíció végének címe.
1E6-1E9	Duplaszó, mely a harmadik partíció kezdetének logikai szektorcímét tartalmazza.
1EA-1ED	Duplaszó, mely a harmadik partíció méretét tartalmazza.
1EE	A negyedik partíció rendszertöltést jelző byte-ja.
1EF-1F1	A negyedik partíció kezdetének címe a lemezen.
1F2	A negyedik partíció rendszer jelző byte.
1F3-1F5	A negyedik partíció végének címe.
1F6-1F9	Duplaszó, mely a negyedik partíció kezdetének logikai szektorcímét tartalmazza.
1FA-1FD	Duplaszó, mely a negyedik partíció méretét tartalmazza.
1FE-1FF	Két byte-os jelző, melynek értéke: 55 AA. Azt jelzi, hogy a szektor a partíciós táblát tartalmazza.

14.1.2 A file leíró tábla (FAT)

A DOS a lemezek foglaltságát az úgynevezett file leíró tábla (File Allocation Table, röviden FAT) segítségével tartja nyilván. Ez a tábla tartalmazza az egy file-hoz tartozó területek láncát is.

A DOS az adatokat a lemezre logikai blokkonként (a DOS szakirodalomban: cluster) viszi ki, és így is tartja nyilván. Egy logikai blokk meghatározott számú (cserélhető lemeznél 1 vagy 2, Winchester-nél 4 vagy 8) szektorból áll.

A FAT 12, illetve 4085-nél több logikai blokkot tartalmazó lemezeknél 16 bites bejegyzéseket tartalmaz. E bejegyzések és a logikai blokkok között egy-egyértelmű megfeleltetés van. Az egyes bejegyzésekhez a velük egyező sorszámú logikai blokk tartozik. Az operációs rendszer a lemezre íráskor megkeresi a FAT-ban az első nem foglalt bejegyzést és az annak megfelelő blokk területére fog írni. Egyidejűleg a FAT-ban bejegyzi a kiírás folytán előállott változásokat.

A FAT első két bejegyzése nincs blokkhoz rendelve. Ezeken a területeken ugyanis a boot rekord, majd a FAT két másolata és a tartalomjegyzék helyezkedik el. A FAT első byte-ján a lemez típusát jelző byte található. A típusleíró byte felépítése bitekre lebontva:

7-3. bit értéke kötelezően 1.
2. bit = 0 az adathordozó nem cserélhető
       = 1 az adathordozó cserélhető

1. bit = 0 az adathordozó nem 8 szektoros
= 1 az adathordozó 8 szektoros
0. bit = 0 egyoldalas lemez
= 1 kétoldalas lemez.

A fenti értelmezés nem hagyott azonosító bit-et a lemezek minden jellemzőjére, pl. méret, felírási sűrűség stb. A PC-DOS 3.30-as verziója a következő standard lemezek használatát biztosítja:

lemez mérete(")	5-1/4	5-1/4	5-1/4	5-1/4	8	8	8
sáv/oldal	40	40	40	40	77	77	77
byte/szektor	512	512	512	512	128	128	128
szektor/blokk	2	2	1	1	4	4	1
fenntartott szektorok	1	1	1	1	1	4	1
FAT-ok száma	2	2	2	2	2	2	2
bejegyzések sz. a főszótárban	64	64	112	112	68	68	192
szektorok száma	320	360	640	720	2002	2002	616
típusleíró byte	FE	FC	FF	FD	FE	FD	FE
szektor/FAT	1	2	1	2	6	6	2
szektor/sáv	8	9	8	9	26	26	8
fejek száma	1	1	2	2	1	1	2
rejtett szektorok	0	0	0	0	0	0	0

lemez mérete(")	5-1/4 3-1/2	5-1/4 3-1/2	5-1/4 3-1/2	5-1/4 3-1/2	5-1/4
sáv/oldal	80	80	80	80	80
byte/szektor	512	512	512	512	512
szektor/blokk	2	2	2	2	4
fenntartott szektorok	1	1	1	1	1
FAT-ok száma	2	2	2	2	2
bejegyzések száma a főszótárban	112	112	112	112	224
szektorok száma	640	1280	720	1440	2400
típusleíró byte	FA	FB	F8	F9	F9
szektor/FAT	1	2	2	3	7
szektor/sáv	8	8	9	9	15
fejek száma	1	2	1	2	2
rejtett szektorok	0	0	0	0	0

A többi bejegyzés tartalma szerint, a következő információkat hordozhatja:

(F)FF8-(F)FFF	A file utolsó blokkját jelzi. (Általában csak az (F)FFF érték használatos.)
(F)FF0-(F)FF7	A hozzá tartozó blokk foglalt, vagy hibás. A fizikai hibás blokkot az (F)FF7 érték jelzi.
(0)000	A bejegyzéshez tartozó blokk szabad, felhasználható.

(n)nnn	Ahol (n)nnn egy 1-nél nagyobb, a blokkok maximális számánál kisebb érték. Azt mutatja, hogy a file folytatása hányadik blokkban található. (A blokk egy file részére foglalt.)

A file első blokkjának sorszáma a file-nak megfelelő tartalomjegyzék bejegyzésben található. A fentieknek megfelelően pedig a FAT e sorszámmal egyező sorszámú bejegyzése tartalmazza a file következő blokkjának sorszámát. A megfelelő FAT bejegyzés megint mutatja a folytatást, míg az utolsó logikai blokkhoz tartozó bejegyzésben (F)FFF érték található.

Megjegyzések:

1. A 12 bites bejegyzések folytonosan helyezkednek el, mégpedig úgy, hogy minden 3. byte alsó négy bitje az előtte lévő byte-tal alkot egy számot, és ennek a számnak a magas helyiértékű részét képezi, a felső négy bitje pedig az utána következő byte-tal alkot egy számot, és ennek a számnak az alacsony helyiértékű részét képezi. Például, ha a FAT első hat byte-ján a következő értékek találhatók:

1.	2.	3.	4.	5.	6.
FD	FF	FF	03	F0	FF

akkor ezek a következő bejegyzéseket jelentik:

1. bejegyzés = F FD
2. bejegyzés = FF F
3. bejegyzés = 0 03
4. bejegyzés = FF F

2. A FAT mindig a boot rekord után helyezkedik el, általában két példányban. A mérete, a másolatok száma, és a blokkméret a boot rekord-ban megtalálható (lásd előző szakasz).

3. Fix lemezeknél a partíció táblában van jelezve, hogy egy bejegyzés 12 vagy 16 bitből áll-e.

4. A 12 bites bejegyzések esetén úgy kaphatjuk meg egy blokk helyét a FAT-ban, hogy a blokk sorszámát (a sorszámozás 0-tól indul) beszorozzuk 1.5-tel. Ha az eredmény egész szám, akkor az így kapott sorszámú byte-on, és a rákövetkező byte alsó négy bitjén található a bejegyzés. Tört eredmény esetén a felkerekített érték a teljes, a lefelé kerekített érték pedig a fél byte helyére mutat.

5. 16 bites bejegyzések esetén a logikai rekord sorszámát 2-vel kell beszorozni, és akkor megkapjuk a rekordhoz tartozó bejegyzés kezdő byte-ját.

6. Ha a rekord logikai sorszámából meg akarjuk kapni a rekord relatív szektorcímét (például a 26-os vagy 25-ös DOS interrupt híváshoz, vagy a DEBUG használatához), akkor a következőképpen járjunk el:
   - vonjunk le a blokkszámból 2-t,
   - szorozzuk meg az eredményt az egy blokkban található szektorok számával,
   - adjuk hozzá az adatterület kezdetének relatív szektorcímét.

14.1.3 A tartalomjegyzék (directory) felépítése, altartalomjegyzékek

A lemezeken a file leíró táblát követően helyezkedik el a főtartalomjegyzék, melyet a FORMAT parancs mindig kialakít. A tartalomjegyzékben a bejegyzések száma korlátozott, a maximális értéket a boot rekord tartalmazza.

Az altartalomjegyzékek a lemezen speciális file-ként helyezkednek el.

Mind a főtartalomjegyzékben, mind az altartalomjegyzékben azonos formátumú bejegyzések tartják nyilván a file-okat.

A tartalomjegyzékekben egy bejegyzés 32 (hexadecimális 20) byte-ból áll és a következő a felépítése:

Hely byte **Tartalom**

0-7 File-név. Ha a 0-dik byte az itt következő értékek valamelyikét veszi fel, akkor a byte a bejegyzés státuszát mutatja és nem file-nevet jelent.

A státusz értékek a következők lehetnek:

00 = Olyan bejegyzés, melyet a rendszer eddig még sohasem használt fel. (Az ilyen bejegyzés nyilvántartása a könyvtárban való keresést gyorsítja meg.)

E5 = A bejegyzést a rendszer már felhasználta, de azt a file-t, amelyet azonosított, már törölte.

05 = Jelzi, hogy a file-név első karaktere E5H.

2E = A bejegyzés nem file-t, hanem alkönyvtárat azonosít. Ha a következő byte értéke szintén 2E, akkor ez a könyvtár nem a gyökér, hanem van "őse". (Ekkor a blokkszám mező, amelynek leírása itt majd következik, a könyvtár ősének kezdő blokkszámát adja, vagy értéke 0000, ekkor pedig az ős a gyökér könyvtár.)

Ha a 0-dik byte értéke nem a fentiek valamelyike, akkor az egyszerűen a file-név első karaktere. (Ha a file-név nem tölti ki a rendelkezésre álló 8 karakternyi helyet, a rendszer balra igazítja.)

8-0A A file-kiterjesztés.

0B A file attributum jelző byte-ja. Egyrészt speciális file-ok megkülönböztetésére, másrészt a file-műveletek sikeres befejezésének ellenőrzésére szolgál. A byte egyes bitjeinek jelentése:

bit **Jelentés**

5. Az úgynevezett archív bit. Ha értéke 1, akkor a file az utolsó írásra való megnyitást követően le is lett zárva. A BACKUP és a RESTORE használja fel ezt a bitet, ebből tudják meg, hogy az utolsó mentés óta változott-e a file. Ezt a bitet a rendszer az attributum byte bármely más bitjével együtt is tudja

használni, azaz ettől függetlenül bármely másik bit is használható. Ha csak ez a bit van beállítva, a byte értéke hexadecimális 20.

4. Altartalomjegyzék azonosító bit. Ha értéke 1, akkor a bejegyzés egy altartalomjegyzéket azonosít, így a normál file műveletekben (REN, DEL, COPY stb.) nem vesz részt.

Ez a bit csak a rejtettséget jelzővel kombinálható, így ha az attributum byte alkönyvtárat jelez, értéke hexadecimális 10 vagy 12.

3. Kötet (volume) azonosító jelzése. Ha a bit értéke 1, akkor a bejegyzéshez nem file tartozik, hanem az első 11 byte-jával egy egész lemezt azonosít. Ilyen bejegyzés a /V opcióval indított formattálással, illetve a LABEL paranccsal hozható létre.

Ez a bit nem kombinálható a többivel, így beállítása esetén az attributum byte értéke 8.

Csak a főtartalomjegyzékben fordulhat elő. Ha több ilyen attributumú bejegyzés található, akkor a rendszer az elsőt tekinti kötet-azonosítónak.

2. Rendszer-file jelzése. Ha a bit értéke 1, akkor a file rendszer-file. Ez azt jelenti, hogy a normál DIR parancs nem kezeli, file-műveletekben nem vesz részt.

1. Rejtett (hidden) file jelzése. Ha a bit értéke 1, akkor a file rejtett file. A DIR műveletekben nem vesz részt, de csoportnévvel DOS parancsokból is elérhető.

0. Csak olvasható (read-only) file jelzése. Ha a bit értéke 1, akkor a bejegyzéshez tartozó file nem írható/módosítható.

Megjegyzések:

1. Ha egyszerre több bit is be van kapcsolva, akkor az egyes bitek magyarázatánál megadott byte értékeket össze kell adni. Például egy lezárt, csak olvasható, rejtett, rendszer-file attributum byte-jának értéke 20 + 4 + 2 + 1 = 27 hexadecimálisan.

2. A 43-as funkcióhívással a következő attributumok megváltoztathatók:

   - csak olvasható file,
   - rejtett file,
   - archív bit.

12-21 A PC-DOS számára fenntartott byte-ok.

22-23 A file létrehozásának, illetőleg utolsó aktualizálásának (update) időpontját tartalmazza, a következő felépítésben:

```
              |     23. byte    |     22. byte    |
   bitek       F E D C B A 9 8 7 6 5 4 3 2 1 0
   tartalom    o o o o o p p p p p p m m m m m
```

ahol

- óóóóó egy 0-23 közötti bináris szám, amely jelzi az órákat
- pppppp egy 0-59 közötti bináris szám, amely jelzi a perceket
- mmmmm egy 0-30 közötti bináris szám, amely jelzi a másodperceket, kettesével növelve.

24-25 A file létrehozásának, illetőleg utolsó aktualizálásának (update) dátumát tartalmazza a következő felépítésben:

```
            |  25. byte    |   24. byte   |
bitek        F E D C B A 9 8 7 6 5 4 3 2 1 0
tartalom     e e e e e e e h h h h n n n n n
```

ahol

• éééééé	egy 0-119 közötti bináris szám, amely az évet jelenti (1980-2099)
• hhhh	egy 1-12 közötti bináris szám, amely a hónapot jelzi
• nnnnn	egy 1-31 közötti bináris szám, amely a napot mutatja.

26-27 A file kezdőblokkjának mutatója, azaz a file első logikai blokkjának blokkszáma. Szó formában van tárolva a file első logikai blokkjának (cluster) címe.

Az adatok számára rendelkezésre álló terület mind a fix, mint a cserélhető lemezeken a 2. blokktól kezdődik.

28-31 A file mérete byte-okban. Duplaszó formátumban előjel nélküli bináris számként tárolódik a file mérete.

14.1.4 Kiterjesztett DOS partíció felépítése

A 3.30-as DOS verzió egyik új eleme a kiterjesztett DOS partíciók használata. Ez lehetővé teszi, hogy a fix lemez(ek)en az ún. elsődleges DOS partíció mellett egy másik, kiterjesztett DOS partíciót is létrehozzunk. Mi több, ennek a nagysága meghaladhatja a 32 Mbyte-ot is.

A kiterjesztett DOS partíciót a rendszer kötetekre osztja, s egy-egy kötethez logikai egységet rendel. Fizikailag a DOS a köteteket, mint fix lemezek DOS partícióit kezeli, azzal a különbséggel, hogy ezekből nem tud rendszert tölteni, bár a partíciós táblában tölthető bejegyzés szerepel. A kiterjesztett DOS partícióban levő boot-rekord (az ún. **kiterjesztett boot rekord**) szerkezete megfelel egy 'igazi' boot rekordnak, az alábbi eltérésekkel:

A **rendszer-jelző byte** értéke az alábbiakat jelenti:

- 00 Ehhez a bejegyzéshez nem tartozik lemezterület
- 01 12 bites DOS partíció
- 04 16 bites DOS partíció
- 05 Ez a bejegyzés a következő kiterjesztett boot rekord helyét jelzi
- 06 Fenntartott

A 01,04 és 06 rendszer-jelző byte-ok esetén a partíció kezdetét és végét jelző három-három byte a kötet megfelelő adatait tartalmazza és a kiterjesztett DOS partíció elejétől számítandó. Ha a 05-ös rendszer-jelző byte-ot használjuk, akkor a három-három byte a következő kötet adatait tartalmazza.

A logikai rekordszám a kötet elejétől mérendő, míg a partícióban levő szektorok száma a logikai egységhez tartozó szektorok számát jelenti.

Ettől eltérően, ha a rendszer-jelző byte 05, akkor a logikai szektorszám az egész kiterjesztett DOS partíció elejétől értendő, és a következő kiterjesztett boot rekord helyét mutatja. A szektorszámot tartalmazó duplaszónak ilyenkor nincs jelentése és 0-val kell feltölteni.

A kiterjesztett boot rekord utolsó két byte-ja az 55AAH számot tartalmazza, jelezve, hogy egy 'igazi' boot rekordról van szó.

14.2 A felhasználói perifériakezelő program (handler, driver)

A DOS perifériakezelését ismertető fejezetben már említettük, hogy a DOS lehetőséget biztosít a felhasználónak, hogy saját perifériakezelő rutint használhasson. A megfelelően elkészített felhasználói perifériakezelő rutin beilleszthető az operációs rendszerbe, azaz azt a gépet használók a beillesztés után úgy érzékelik, mint a DOS szerves részét.

14.2.1 A perifériakezelő program felépítése

A perifériakezelő (vagy driver) speciális memóriaképű program. COM, EXE vagy SYS kiterjesztésű file tartalmazhatja. Egy file-ban több, egymástól független perifériakezelő is elhelyezhető.

A file-beli egyes periféria-kezelő programok elején egy-egy táblázat helyezkedik el - az úgynevezett **perifériakezelő fejléc (device header)** -, amely tartalmazza a programhoz tartozó belépési címeket, a kezelendő berendezés típusát, nevét, a következő drivert megelőző fejléc kezdőcímét. (A fejlécet részletezzük majd.) A fejlécet követi a tényleges perifériakezelő program.

A perifériakezelőt leíró programnak nem kell a PSP részére maga előtt hexadecimális 100 byte-ot fenntartania, azaz assemblerben való megírása esetén nem szabad az ORG 100H direktívával kezdődnie a programnak. A perifériakezelőt ugyanis a DOS egyszerű file-ként, PSP nélkül tölti be.

A DOS a felhasználói perifériakezelő létezéséről a CONFIG.SYS file-ból szerez tudomást, ugyanis, ha saját perifériakezelőt akarunk alkalmazni, a CONFIG.SYS file-nak tartalmaznia kell egy DEVICE=*file-név* paramétert. A file-név a felhasználói perifériakezelőt tartalmazó file neve. Természetesen a DOS csak új rendszertöltés után fogja a perifériakezelőt felismerni, nem pedig a CONFIG.SYS átírása után. (A témához kapcsolódó információk találhatók a "Rendszer konfigurálása" és a "PC-DOS belső struktúrája" című, 11. és 13. fejezetekben.)

A perifériakezelő fejlécének előírt szerkezete a következő:

Cím byte	**Adattartalom**
0-3	Duplaszó, amely a file-on belül rámutat a következő perifériakezelő program fejlécére. Ha ez a program a file-ban az utolsó, akkor az első két byte értéke FFFF. Ha több program van egy file-ban, de azok egy szegmensen belül helyezkednek el, akkor a második két byte értéke FFFF.
4-5	Az úgynevezett attributum-szó, ez a berendezés jellemzőit írja le. Az attributum-szó egyes bit-jeinek jelentése a következő: 15. Ha karakteres periféria, akkor 1, különben 0. 14. Ha a DOS a perifériára elfogadja a 44-es funkcióhívás 3., 4. és 5. alparancsát, azaz támogatja az IOCTL-t (perifériát vezérlő információ), akkor 1, különben 0. 13. Nem IBM formátumú blokkos periféria esetén 1, IBM formátumú blokkos periféria esetén 0. 12. Fenntartott. 11. Ha az egység megérti az OPEN/CLOSE parancsot 1, különben 0. 10.-7. Fenntartott. 6. Ha az egység megérti a 3.2 és 3.3 parancsokat - támogatja a Get/Set Logical Device-t (lemez logikai/fizikai hozzárendelését) - akkor 1, különben 0. 5.-4. Fenntartott. 3. Ha a periféria a használt óraeszköz, akkor 1, különben 0. 2. Ha a periféria az aktuális NUL berendezés, - azaz erre tetszőleges output-ot kiküldhetünk anélkül, hogy bármi hatása lenne -, akkor 1. 1. Ha a periféria az aktuális konzol output, akkor 1, különben 0. 0. Ha a periféria az aktuális konzol input, akkor 1, különben 0.
6-7	Szó, amely annak a segédrutinnak a belépési címe, ami átveszi a perifériakezelő részére kiadott parancsblokkot (a parancsblokk leírása később következik).
8-9	Szó, amely a parancsot végrehajtó segédrutin belépési címe.
0A-11	8 byte, amely a berendezés nevét tartalmazza.

Megjegyzés:

1. A 44-es funkció segítségével a perifériára vezérlő karakterek küldhetők, illetőleg onnan vezérlő karakterek fogadhatók. Ezen a módon kezelhető például a baud rate, a stop bit, a változó formátumú rekord hossza stb. A funkció használatát részletesen ismerteti a funkcióhívásokat tartalmazó fejezet megfelelő szakasza.

2. A fejrész utolsó 8 byte-ja a névmező, amely karakteres perifériáknál egy maximum ilyen hosszú, a DOS szabályainak megfelelő nevet tartalmaz. Ha a név rövidebb, mint 8 karakter, akkor a mező elején (0A. azaz decimális 10. byte-on) kell kezdődnie, és a végén lévő szabad byte-okat szóközzel kell kitölteni.

   Blokkos perifériánál az első byte-nak az alegységek (unit) számát kell tartalmaznia. Ezek száma és a már lefoglalt egységek alapján rendel betűjelet a perifériákhoz a DOS. A továbbiakban az egységre a parancsokban az így kapott betűvel hivatkozhatunk.

   Amennyiben a névmező létező periférianevet tartalmaz, úgy a DOS a saját perifériakezelője helyett a felhasználóét fogja használni.

14.2.2 A perifériakezelő programokkal kapcsolatos tudnivalók összefoglalása és kiegészítése

Mint már említettük, a perifériakezelő program részei:

- a perifériakezelő fejléce,
- a parancsblokk átvevésére szolgáló segédrutin (ennek belépési címét a fejléc 6. és 7. byte-ja kell, hogy tartalmazza),
- a konkrét feladatot végrehajtó segédrutin (ennek belépési címét a fejléc 8. és 9. byte-ja kell, hogy tartalmazza).

Ha a CONFIG.SYS-ben elvégeztük az említett (DEVICE = file-név) javítást, akkor, új bekapcsolás vagy rendszertöltés után, a DOS betölti a file-t, amely tartalmazza a driverünket. Ezután minden file-számos file-megnyitásnál ellenőrzi, hogy a megnyitandó file-név nem egyezik-e valamely perifériakezelő névvel. Ha igen, akkor erre a file-ra való hivatkozások a DOS-ból kiváltják a driver-ünk megfelelő paranccsal való meghívásait. A driver meghívása két lépésből áll. Az elsőben a DOS felépít egy megfelelő parancsblokkot és meghívja a driver parancsblokk kezelő rutinját. E híváskor a DOS átadja a rutinnak ES:BX-ben az általa felépített fejléc címét. Ezen a területen a "parancskód"-nak nevezett byte-ban közli, hogy milyen feladat ellátását várja a driver-től, és kiegészíti a fejlécet az adott parancskódot jellemző, további parancsblokk részlettel.

A parancsblokk-átvevő segédrutint nekünk kell megírnunk. Minden esetre el kell tennünk a kapott fejléc-címet, és a DOS-tól kapott parancskód szerint kitöltjük a fejléc adatait. Az is a mi feladatunk, hogy a gép regisztereit a rutin kezdetén kimentsük, és mielőtt a DOS-nak visszaadjuk a vezérlést, visszatöltsük. Ezt a másik segédrutinnál is meg kell tennünk.

Lehetőség van arra is az első segédrutinnál, hogy a más gépeknél "interrupt driver" feladatoknak nevezett perifériakezelési módszereket beprogramozzunk, például sorba állíthatjuk az egyazon perifériára vonatkozó kéréseket stb.

Ha minden mentési, ki- és visszatöltési feladatot elvégeztünk, visszaadjuk a DOS-nak a vezérlést.

Erre a DOS azonnal meghívja ismét a handler-ünket, ezúttal a második segédrutin belépési címére adva a vezérlést. Most kell ellátnunk azt a feladatot, amit a DOS a parancskódban közölt velünk.

Mielőtt ismertetnénk a parancsblokk felépítését és a lehetséges parancskódokat, néhány dologra felhívjuk a figyelmet:

A DOS csak a regiszterek kimentésére biztosít veremterületet. Ha többre van szükségünk, azt magunknak kell lefoglalnunk és kezelnünk.

A DOS FAR CALL-lal hívja a rutinjainkat, így a rutinjainknak is ennek megfelelően kell visszatérniük. Mindkét segédrutint a fejléccel azonos szegmensben kell elhelyezni, és azoknak távoli RET utasítással (RETF azaz FAR RET) kell végződniük.

Ha több perifériakezelő található egy file-ban, ezeknek lehet közös segédrutinjuk, például a parancsblokk átvételére.

A parancsblokk-átadó segédrutin a fejléc címét a DOS-tól kapja, de a feladatot végrehajtó segédrutinnak már ő továbbítja, a feladatot kijelölő parancskóddal és egyéb információval együtt.

A perifériakezelő programoktól a DOS az itt következő feladatok ellátását várja el. (Az egyes feladatokhoz tartozó parancsblokkot a megfelelő feladatnál ismertetjük.)

DOS-tól kapott parancskód	**Az ellátandó feladat**
0	inicializálás
1	adathordozó csere megtörténtének ellenőrzése
2	BPB (Bios Paraméter Blokk, blokkos periférialeíró táblája) kitöltése (lásd a részletes ismertetést)
3	input/output vezérlő információ lekérdezése
4	olvasás perifériáról
5	karakteres periféria állapotának lekérdezése, az átviteli puffer tartalmának megőrzésével
6	karakteres periféria input puffer állapotának lekérdezése
7	karakteres periféria input puffer tartalmának törlése
8	írás perifériára
9	perifériára írás, írás utáni ellenőrzéssel
10	karakteres periféria output puffer állapotának ellenőrzése
11	karakteres periféria output puffer tartalmának törlése
12	perifériát vezérlő információk beállítása
13	periféria megnyitása (csak a 3.x verziókban, s ha a periféria attributum szavában a 11. bit magas)
14	periféria lezárása (csak a 3.x verziókban, s ha a periféria attributum szavában a 11. bit magas)
15	lemez cserélhetőségének lekérdezése (csak a 3.x verziókban, s ha a periféria attributum szavában a 11. bit magas)
16	írás perifériára míg foglalt nem lesz (csak a 3.10-es verzióktól)

19	vezérlő karaktersorozat elküldése (csak a 3.10-es verzióktól, s csak akkor, ha a periféria attributum szavában a 13. bit magas.)
23	lemez logikai-fizikai hozzárendelésének lekérdezése (csak a 3.2-es verzióktól, s ha a periféria attributum szavának 6. bitje magas)
24	lemez logikai-fizikai hozzárendelésének beállítása (csak a 3.2-es verzióktól, s ha a periféria attributum szavának 6. bitje magas)

A felhasználói perifériakezelő programot úgy kell megírni, hogy a fenti parancsokat fogadja a 14.2.4 részben részletezett formátumban, és a parancsok hatására megfelelően működjön.

14.2.3 A parancsblokk és fejlécének felépítése

A perifériakezelő program és a DOS operációs rendszer, mint említettük, egy úgynevezett parancsblokk segítségével kommunikál egymással. A parancsblokk első 13 byte-ja egységesen ugyanaz minden parancs esetében, ezt neveztük parancs fejlécnek (Request Header). Amikor a DOS meghívja a perifériakezelő program parancsblokk átvevő rutinját, akkor az ES:BX mutat a parancs fejléc első byte-jára. A rutinnak kell gondoskodni a mutató elmentéséről.

A fejléc szerkezete a következő:

Helyfoglalás (byte-ban)	**Adattartalom**
1	A teljes parancsblokk hossza byte-okban kifejezve.
1	Blokkos periféria esetén a használandó alegység száma (például, ha a program 2 egységet kezel, e szám lehetséges értékei 0 és 1). (Karakteres periféria esetén ez a byte nem értelmezett.)
1	Parancskód. Ez fejezi ki a perifériakezelő program feladatát. A lehetséges értékeket a következő szakasz részletezi.
2	Állapotjelző szó. Tartalmát alább részletezzük.
8	Fenntartott terület a DOS számára.

Az állapotjelző szó tartalma:

bit	**Jelentés**
15	Hiba fellépésének jelzése. Értéke, ha nem történt hiba, 0, egyébként 1, és ekkor a hibakódot az állapotjelző szó alacsonyabb helyiértékű része tartalmazza, mint az majd következik.
14-10	A DOS számára fenntartott terület.
9	Úgynevezett foglaltsági bit. Bizonyos perifériakezelési feladatoknál a periféria foglaltságát vagy a háttértár cserélhetőségét jelzi. A következő szakaszban térünk erre vissza, a perifériakezelő programok feladatainak ismertetésénél.
8	A művelet befejezettségét jelző bit. Értéke akkor lesz 1, ha a művelet befejeződött, egyébként 0.

7-0 Hibakód. Ha a 15. bit 1, akkor a hiba kódját tartalmazzák, az alábbi táblázat szerint:

Hibakód	**Hiba leírása**
0	Íráskísérlet írásvédett lemezre.
1	Ismeretlen periféria.
2	A periféria nincs műveletre kész állapotban.
3	Ismeretlen fajta perifériakezelő parancs (lásd a következő szakaszt).
4	CRC hiba.
5	A parancsblokk hossza hibás.
6	Keresési (seek) hiba.
7	Ismeretlen adathordozó típus.
8	Nem találja az adott szektort.
9	A papír kifogyott a nyomtatóból.
0A	Íráshiba.
0B	Olvasási hiba.
0C	Nem azonosítható hiba.
0D	Fenntartott.
0E	Fenntartott.
0F	Nem megengedett lemezcsere.

14.2.4 A perifériakezelőt vezérlő parancsok részletes leírása

Inicializálás

Parancskód: 0

A parancsblokk felépítése:

Byte	**Jelentés**
0-0C	A parancsblokk fejléce.
0D	Blokkos periféria esetén a handler által kezelt egységek, száma karakteres perifériáknál nincs értelmezve.
0E-11	A perifériakezelő utáni első szabad memóriacím értéke. Típusa: duplaszó.
12-15	Blokkos periféria esetén a BPB vektor, azaz a blokkos egységek leíró tábláit kijelölő címtáblázat címe, karakteres perifériánál nincs értelmezve. (A BPB ismertetése megtalálható a 14.1.1 részben.) Típusa: duplaszó.
16	A periféria egységszáma.

Feladata:

A DOS a perifériakezelő rutin betöltésekor egyszer hívja meg ezt a parancsot. A rutinnak be kell állítania az állapotjelző szót, a program által szabadon hagyott első memóriacímet, a kezelt egységek számát, valamint lemezes meghajtó esetén a BPB táblázat helyét megmutató címet. Ebben a programrészben kell a driver-nek minden olyan kezdeti feladatot ellátnia, amelyet a periféria megkíván.

Blokkos eszközök esetén a 12-15. byte-okon levő mutató a rutin meghívásakor a CONFIG.SYS file-ban levő DEVICE= utáni első karakterre mutat.Ez lehetővé teszi a paraméterek átvételét.

Adathordozó csere megtörténtének ellenőrzése

Parancskód: 1

A parancsblokk felépítése:

Byte	**Jelentés**
0-0C	A parancsblokk fejléce.
0D	Az adathordozó típusa - a DOS adja meg. A típusleíró byte-ot a 2. parancsnál ismertetjük.
0E	Csere állapotának jelzése: -1 az adathordozót cserélték 0 a csere ténye nem megállapítható 1 csere nem volt.
0F-12	Csere esetén az adathordozó kötetazonosítójára (volume identifier) mutató duplaszó.

Feladata:

A parancs hatására meg kell állapítani, hogy történt-e a kérdezett egységben csere. A lekérdezés eredménye a következő 4 lehetőség valamelyike:

- nem történt csere,
- csere történt,
- a csere nem megállapítható,
- hiba lépett fel.

A megállapított eredménynek megfelelően be kell állítani a fejlécben a visszatérési kódot, a csere állapotát jelző byte-ot, továbbá csere esetén az új adathordozó kötetazonosítójára mutató pointert.

Megjegyzés:

1. Ha a DOS csere esetén megállapítja, hogy az hibás, akkor 0F kódú hibát jelez a kérdéses egységre.
2. Ha csere történt, de az adathordozó nem tartalmazott kötetazonosítót, akkor az erre mutató pointernek (0F-12) egy "NO NAME",0 string-re kell mutatnia.
3. A DOS által használt adathordozók típusai a következő, a 2-es parancsnál találhatók.
4. A DOS 3.2-esnél régebbi verziói az adathordozó típus byte-ját használták a lemez struktúrájának azonosítására. Ha csere történt, de a visszaadott BPB típusleíró byte-ja ugyanaz volt, akkor a DOS azt tételezte fel, hogy a cserélt lemez felépítése ugyanaz.

BPB (BIOS Parameter Blokk) kitöltése

Parancskód: 2

A parancsblokk felépítése:

Byte	**Jelentés**
0-0C	A parancsblokk fejléce.
0D	Az adathordozó típusa. A programozó adja meg. (Lásd a megjegyzést.)
0E-11	A DOS által a handler részére átengedett puffer címe (duplaszó).
12-15	A perifériakezelő által összeállított BPB (egység leíró tábla) címe (duplaszó).

Feladata:

Az adathordozón található boot rekord alapján össze kell állítania az egységben található adathordozó a valóságnak megfelelő típus leíró byte-ját, és az egység leíró táblát. A típusleíró byte-ot és az egységleíró tábla címét a perifériakezelő a parancsblokkban visszaadja a DOS-nak.

Megjegyzés:

Az adathordozó típusleíró byte megtalálható a lemezeken a FAT 0. byte-jaként, és felépítése a következő:

7-3. bitek	Értéke kötelezően 1.
2. bit	Ha az adathordozó cserélhető, akkor 1, különben 0.
1. bit	Ha az adathordozó 8 szektoros, akkor 1, különben 0.
0. bit	Ha kétoldalas a lemez, akkor 1, különben 0.

A fenti értelmezés nem hagyott azonosító bit-et a lemezek minden jellemzőjére (például átmérő, felírási sűrűség). A létező azonosító byte-ok a fejezet elején találhatók meg.

Perifériavezérlő input/output parancsok

Parancskód: 3,4,8,9,12,16

A parancsblokk felépítése:

Byte	**Jelentés**
0-0C	A parancsblokk fejléce.
0D	Az adathordozó típusa - a DOS adja meg.
0E-11	Duplaszó, amely az átviteli puffer címét tartalmazza.
12-13	Az átviendő byte-ok, illetve szektorok számát tartalmazza. A művelet befejezése után a periféria-kezelőnek be kell írnia a ténylegesen átvitt egységek számát.
14-15	Szó, amely blokkos eszköz esetén a kezdő szektor sorszámát tartalmazza.
16-19	Duplaszó, amelyben 0FH hibakódú hiba (nem megengedett lemezcsere) esetén be kell állítani annak a karaktersorozatnak a címét, amely a helyes kötetazonosítót tartalmazza.

Input/output vezérlőinformáció lekérdezése.
Parancskód: 3
Feladata:

A perifériakezelő programnak át kell adnia a parancsblokk adatainak megfelelő beállítása segítségével a periféria vonatkozó vezérlő információt. További információk találhatók erre vonatkozóan a 44-es funkció leírásánál.

Olvasás a perifériáról
Parancskód: 4
Feladata:

A parancsblokkban megadott számú byte-ot (karakteres periféria), illetőleg szektort (blokkos periféria esetén) kell a handler-nek beolvasnia a perifériáról.

Írás a perifériára
Parancskód: 8
Feladata:

A feladat ugyanaz, mint a 4. parancsnál, csak írni kell.

Perifériára írás, írás utáni ellenőrzéssel
Parancskód: 9
Feladata:

Ugyanaz, mint az előző parancs, de az írás sikerességét utólag le kell ellenőrizni és ennek megfelelő értékű állapotjelző szót kell a parancsblokk fejléc részében visszaadni.

A perifériát vezérlő információk beállítása
Parancskód: 12
Feladata:

A perifériakezelő program átveszi a 44-es funkciónál leírtak szerint a kívánt vezérlő információkat, és a továbbiakban ennek megfelelően kell működnie.

Írás a perifériára, míg foglalt
Parancskód: 16
Feladata:

A 16. parancs biztosítja, hogy ki lehessen használni a legtöbb nyomtatónál rendelkezésre álló pufferolási lehetőséget. A pufferolási lehetőségek ismeretében nem kell megvárni, míg a nyomtató jelzi, hogy kész, hanem rögtön vissza lehet adni a vezérlést a DOS-nak. (Ha az attribútum szóban beállítottuk, hogy a periféria megérti a 3.2-es parancsokat, csak abban az esetben használhatjuk ezt a parancsot.)

Karakteres periféria olvasása az átviteli puffer tartalmának megőrzésével
Parancskód: 5

A parancsblokk felépítése:

Byte	Jelentés
0-0C	A parancsblokk fejléce.
0D	A perifériáról olvasott byte-ot ide teszi a handler.

Feladata:

A handler-nek a karakteres periféria pufferének állapotát kell a PC-DOS felé visszajeleznie a következő módon:

- ha a puffer üres, akkor a parancsblokkban az állapotjelző szó foglaltsági bitjét (9. bit) nullára kell állítani,
- ha a puffer adatot tartalmaz, akkor a jelzett bitet 1-re kell állítani, továbbá a pufferből a karaktert be kell rakni a parancsblokkba. A puffer tartalma nem törlődik.

Karakteres periféria input puffer állapotának lekérdezése

Parancskód: 6

A parancsblokk felépítése:

Byte	Jelentés
0-0C	A parancsblokk fejléce.

Feladata:
A parancs megegyezik az 5-ös paranccsal. Az egyetlen eltérés, hogy a parancsblokkban csak az állapotot kell visszaadni, a pufferban lévő karaktert nem.

Megjegyzés:
Azok a perifériák, amelyeknek nincs pufferük, az 5. és a 6. parancsnál is 0 foglaltsági bitet kell hogy visszaadjanak, hogy a DOS ne várakozzék feleslegesen.

Karakteres periféria input puffer tartalmának törlése

Parancskód: 7

A parancsblokk felépítése:

Byte	Jelentés
0-0C	A parancsblokk fejléce.

Feladata:
A perifériakezelőnek törölnie kell az input pufferben található valamennyi karaktert, törölnie kell az összes, a perifériára vonatkozó kérést, amelyet nyilvántartott, és be kell állítania a fejlécben az állapotjelző szót.

Karakteres periféria output puffer állapotának ellenőrzése

Parancskód: 10

A parancsblokk felépítése:

Byte	Jelentés
0-0C	A parancsblokk fejléce.

Feladata:
Ha a handler kész a következő karakter feldolgozására, akkor az állapotjelző szó foglaltsági (9.) bitjét nullára kell állítania, ellenkező esetben 1-re.

Karakteres periféria output puffer tartalmának törlése.
Parancskód: 11

A parancsblokk felépítése:

Byte	Jelentés
0-0C	A parancsblokk fejléce.

Feladata:
A handler-nek törölnie kell az output puffer teljes tartalmát, és be kell állítania az állapotjelző szót.

Adathordozó cserélhetőségének lekérdezése
Parancskód: 15

A parancsblokk felépítése:

Byte	Jelentés
0-0C	A parancsblokk fejléce.

Feladata:
A parancsot csak a DOS 3.xx-es változatai hívják, s azok is csak akkor, ha a periféria-kezelő attributum szavában beállítottuk az OPEN/CLOSE/RM bitet. A parancs végrehajtása után a periféria-kezelőnek jeleznie kell, hogy az adathordozó cserélhető-e vagy sem. Az információt az állapotjelző szó foglaltságot jelző bitjében kell visszadni. Ha magas (1), akkor a lemez nem cserélhető, ha 0, akkor igen.

Vezérlő karaktersorozat átvitele a perifériára
Parancskód: 19

A parancsblokk felépítése:

Byte	Jelentés
0-0C	A parancsblokk fejléce.
0D	Byte: kategória kód.
0E	Byte: függvény kód.
0F-10	Szó: (SI) tartalma.
11-12	Szó: (DI) tartalma.
13-16	Duplaszó: az adatpuffer kezdőcímére mutat.

Feladata:
A parancs a 44-es funkció 2,3,4,5 alfunkcióit hajtja végre az adott perifériára. Az egyes paraméterek leírása ott található meg.

File megnyitása
Parancskód: 13

File lezárása
Parancskód: 14

A parancsblokk felépítése:

Byte	**Jelentés**
0-0C	A parancsblokk fejléce.

A DOS 3.xx-es verziói, ha a beállítottuk az OPEN/CLOSE/RM bitet a periféria attributum szavában, jelzik a periféria kezelőnek, ha a periférián megnyitották/lezárták egy file-t. Lehetőség van ilyenkor valamilyen tevékenység elvégzésére (pl. egy y inicializáló karaktersorozat elküldésére a nyomtatóra), vagy hibát jelezni, ha egynél több file nem lehet nyitva a periférián.

A CON, AUX és PRN perifériák mindig nyitottak (a hozzájuk rendelt file-számok: 0,1,2,3,4)

Logikai-fizikai hozzárendelés lekérdezése
Parancskód: 23

Logikai-fizikai hozzárendelés beállítása
Parancskód: 24

A parancskód felépítése:

Byte	**Jelentés**
0-0C	A parancsblokk fejléce.

Feladata:

A DOS ezt a funkciót csak akkor hívja, ha a periféria attributum szavában a 3.2-es bitet beállítottuk. Ebben az esetben ugyanazt a fizikai eszközt több logikai eszköz is birtokolhatja. A két parancs ezt a hozzárendelést kérdezi le, illetve állítja be.

A DOS a parancsblokk fejlécének az egységszámot tartalmazó részében adja át a logikai egységszámot és ott kéri vissza azt is, hogy kié volt a meghajtó utoljára.

14.2.5 Javaslat gyakorlati példára

Az IBM PC-k kedvelt tartozéka, az AT-k szerves része a "Real Time Clock", azaz a hálózatfüggetlen óra. Ezt a kártyát az átlag felhasználó a TIME és a DATE parancsokkal éri el. Ez speciális periféraként funkcionál.

Amint azt a perifériakezelő fejléc attributum szavának leírásánál mondottuk, az attributum szó 3-as bitje jelzi, ha a periféria az óra. Az óra kezelésére a DOS-ban levő driver teljesen hasonlóan működik, mint a normál karakteres perifériáké. Amennyiben olvasni próbálunk a perifériáról, vagy írni akarunk rá, úgy 6 byte kerül átvitelre. Az első két byte egy szó, amely folyamatosan számolja a napokat 1980. január 1-től kezdve. A harmadik byte a perceket, a negyedik az órákat, az ötödik a századmásodperceket, végül a hatodik a másodperceket tartalmazza.

A megfelelő DOS funkciók segítségével nyissuk meg (3D funkció) a CLOCK$ nevű file-t, és olvassuk le róla (3F funkció) a fenti hat byte-ot. E funkció hívásakor a DOS az előző szakaszban ismertetett parancsok segítségével látja el a funkcióhívásokkal megadott feladatot.

Végül azoknak, akik a gyakorlatban meg akarják ismerni egy felhasználói perifériakezelő rutin szerkezetét, javasoljuk, hogy a DEBUG segítségével vizsgálják meg az ANSI.SYS driver-t. Mint ismeretes, az ANSI.SYS biztosítja, hogy escape szekvenciákkal lehessen a képernyőre írást irányítani, Azért javasoljuk ezt, mert a DOS 2.00 vagy későbbi változatokhoz az ANSI.SYS file biztos, hogy rendelkezésre áll.

A fenti módszernél lényegesen egyszerűbb, ha a 3.00-ás verziótól kezdődően a DOS-hoz adott VDISK.LST nevű file-t tanulmányozzák át, amelyik a VDISK.SYS driver assembly listáját tartalmazza.

15. A DOS memóriakezelése, munkaterületei és vezérlőblokkjai

Ez a fejezet a memória felosztását és kezelését, a DOS-nak a programokról és a file-okról nyilvántartott információit és ezek felhasználási lehetőségét tekinti át.

A memóriatérkép ismertetése után részletezzük a DOS adatterületeit és az alkalmazói programokat kiszolgáló táblázatokat: a PSP-t (Program Segment Prefix) és a programkörnyezetet (environment). Ezt követően a memória kezelésével és nyilvántartásával foglalkozunk. A fejezet második része a perifériakezeléssel kapcsolatos táblázatokat (file leíró blokk, lemezátviteli terület, felhasználói perifériakezelők vezérlőblokkja) foglalja össze.

15.1 A DOS memóriatérképe

A memóriatérkép megmutatja, hogy a számítógép memóriájában milyen sorrendben és milyen címeken helyezkednek el a BIOS (Basic I/O System) és a DOS elemei, valamint a felhasználói programok.

Memóriacím (szegmens:offset)	**Terület megnevezése**
0000: 0000-03FF	megszakítási vektor tábla
0000-007F	BIOS megszakítások (INT 0-1F)
0080-00FF	DOS megszakítások (20-3F)
0100-017F	fenntartott (40-5F)
0180-019F	megszakítások felhasználói programok részére (60-67)
01A0-01FF	nem használt megszakítások (68-7F)
0200-0217	foglalt a BASIC részére (80-85)
0218-03C3	BASIC interpreter által a programok futása alatt használt megszakítások (85-0F0)
03C4-03FF	nem használt megszakítások (0F1-0FF)
0040: 0000-00FF	BIOS adatterület
0050: 0000-00FF	DOS adatterület
0000	képernyőprintelés (hardcopy) jelző; ha tartalma: 0 - képernyőprintelés inaktív vagy sikeresen végetért 1 - képernyőprintelés folyamatban 255 - hiba lépett fel a képernyőprintelés alatt
0001	a BASIC használja
0004	egy meghajtós üzemmód jelző, ha tartalma: 0 - az A meghajtót használták utoljára 1 - a B meghajtót használták utoljára
0050: 0010-0021	a BASIC használja
0022-002F	adatterület lemezmeghajtók inicializálásához
0030-0033	adatterület a MODE parancs részére

XXXX:0000-	a ROM I/O rutinokra épülő alapszintű perifériakezelő (IBMBIO. COM)
YYYY:0000-	a DOS megszakítást kezelő és funkcióhívási rutinjai (IBMDOS. COM)
	DOS pufferek, vezérlő területek és a felhasználói perifériakezelők
ZZZZ:0000-	a parancsfeldolgozó (COMMAND.COM) állandó részei, valamint a DOS 22, 23, 24-es megszakításkezelő rutinjai
VVVV:0000-	külső parancsok vagy segédprogramok (.COM vagy .EXE file-ok) veremtár a .COM programok részére (decimális 256 byte)
WWWW:0000-	a parancsfeldolgozó átmenetileg tárolt részei, azaz parancs értelmező, belső parancsok, batch feldolgozó, külső parancs betöltő

A memóriatérképben szereplő XXXX, YYYY, ZZZZ, VVVV, WWWW szegmenscímek értéke a DOS megvalósításától függően változhat, de mindenképpen egymást követő tárterületeket jelölnek ki.

15.2 A programszegmens prefix (PSP)

Külső parancs vagy felhasználói program indításakor az operációs rendszer a legalacsonyabb című szabad tárterületet használja fel. Az így felhasznált memória a programszegmens. A programszegmens első 256 (hexadecimálisan 100) byte-ján az operációs rendszer egy vezérlőblokkot, az úgynevezett PSP-t (Program Segment Prefix-et) hoz létre. A PSP-t követően helyezkedik el az indítandó program futásra alkalmassá tett kódja. (A 'COM' kiterjesztésű programok tárfüggetlen kódot tartalmaznak, azaz tetszőleges helyre betöltve hibátlanul működnek. Az 'EXE' kiterjesztésű programok ellenben olyan utasításokat is tartalmaznak, amelyek függnek a program kezdőcímétől, ezért a betöltéskor ezeket az utasításokat módosítani kell. Ezt a módosítást az operációs rendszer végzi el. Ez a művelet a relokálás (programáthelyezés, ezt a műveletet az EXE fejléc leírásánál ismertetjük majd).

Az operációs rendszer programindításkor a programszegmens számára lefoglalja a felhasználható teljes tárterületet. Amennyiben a program ekkora területet nem igényel, módosíthatja a szegmens méretét. Ha a program befejezte a futását, a DOS-hoz visszatérve a visszatérés módja szerint vagy megszűnik a programszegmens, vagy megmarad (rezidens kilépés). Utóbbi esetben a későbbiek során a rezidens program újabb betöltés nélkül elindulhat. Ilyen esetekben a gép memóriájában egyidejűleg több programszegmens követi egymást. Normál programbefejezés elérhető INT 20 hívással, vagy a 0-ás, vagy a 4C-s funkcióhívással, továbbá a PSP 0. byte-jára való ugrással (JMP 0). Megjegyezzük, hogy az utóbbi esetben, valamint INT 20 használatakor gondoskodni kell arról, hogy a CS szegmensregiszter a PSP-re mutasson. Ellenkező esetben (azaz ha a PSP nem a CS:0000 címen található), a programbefejezés után nem fog a DOS bejelentkezni, a további munkához rendszertöltés szükséges. Ha azt akarjuk, hogy a program befejezése után is megmaradjon a programszegmens, akkor INT 27-tel vagy a 31-es funkcióhívással kell lezárni.

15.2.1 A PSP felépítése

Mint említettük, a gép memóriájában egyidejűleg több PSP is bent lehet. A tárkezeléshez, a program indításához és befejezéséhez, valamint betöltéséhez,továbbá az operációs rendszerrel való kapcsolathoz szükséges információk a PSP-ben találhatók. Ebből következik, hogy a PSP bizonyos adatait a program felhasználhatja, egyeseket módosíthat is, másokat azonban nem változtathat meg. A 0-5C közötti területet nem szabad átírni.

A PSP a következőket tartalmazza:

Cím byte	**Tartalom**
0-1	A 2 byte tartalma: CD 20. Ez egy INT 20 utasítás. A program erre a címre ugorva befejezheti a futását. (A programbefejezési lehetőségeket lásd az interrupt-ok ismertetésénél.)
2-3	Memória felső határa. Az első nemlétező memóriacím paragrafusban kifejezve. Példánkban ez 3D80. Ez decimálisra átszámolva, és figyelembe véve, hogy egy paragrafus 16 byte, a gép memóriájára 247.808 byte-ot ad.
4	Fenntartott
5-9	A DOS funkciókkal kapcsolatos LONG CALL utasítás.
0A-0B	Program befejezési cím offsetje.
0C-0D	Program befejezési cím szegmenscíme.
0E-0F	<Ctrl-Break> kilépési cím offsetje.
10-11	<Ctrl-Break> kilépési cím szegmenscíme.
12-13	Hibakezelő rutin cím offsetje.
14-15	Hibakezelő rutin cím szegmenscíme.
16-2B	DOS részére foglalt terület.
2C-2D	A programkörnyezet szegmenscíme.
2E-4F	DOS részére foglalt terület.
50-52	Ez a három byte egy, a DOS funkciókat (példánkban CD 21 CB, ami éppen INT21 RET FAR) meghívó utasítást tartalmaz. Így a program a 21-es interrupt-on kívül úgy is hívhat DOS funkciót, hogy AH-ban beállítja a kívánt funkció számát, és FAR CALL szubrutinhívást ad ki erre a címre.
53-5B	DOS részére foglalt terület.
5C-6B	Normál nem megnyitott file-leíró blokk (FCB1).
6C-7F	Második normál nem megnyitott file leíró blokk (FCB2). Az FCB1-et és az FCB2-t a DOS alakítja ki. Ha az adott PSP-t létrehozó DOS parancs file-t vagy file-okat is megad, akkor e paraméterekből kiveszi a file-okra vonatkozó információt és ide beírja, amint a példában is látni fogjuk. Ha a parancs nem ad meg file-t, akkor a meghajtó az aktuális lesz (0), a file-azonosító meg space-szel lesz feltöltve. Ha az FCB1-et megnyitjuk, akkor az FCB2 felülíródik.
80-FF	Lemez átviteli terület (DTA). A parancsfeldolgozó ezt jelöli ki DTA-nak és a 81. byte-tól kezdődően itt tárolja a konzolról beolvasott sor parancsot követő részét, a program indítási paramétert. A program indítási paraméter hossza a 80-as címen található.

A fentieket egyszerűen megnézhetjük a DEBUG utasítás segítségével.

Adjuk ki a DEBUG COMP.COM A:XXX B:YYY parancsot.

A DEBUG utasítás a betöltődése után be fogja tölteni az indítási paraméterén található file-t, azaz COMP.COM-ot. A betöltés után a -DO.100 parancs segítségével megnézhetjük a PSP-t, ami COMP A:XXX B:YYY parancshoz tartozik:

```
0DA0:0000 CD 20 80 3D 00 9A F0 FF-0D F0 42 02 99 0A 70 02  M .=..p..pB...p.
0DA0:0010 99 0A A2 03 34 06 99 0A-01 01 01 00 02 FF FF FF  ..•.4...........
0DA0:0020 FF FF FF FF FF FF FF FF-FF FF FF FF 9B 0D 49 2A  ..............I*
0DA0:0030 99 0A 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0dA0:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0DA0:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 01 58 58 58  MIK..........XXX
0DA0:0060 20 20 20 20 20 20 20 20-00 00 00 00 02 59 59 59          .....YYY
0DA0:0070 20 20 20 20 20 20 20 20-00 00 00 00 34 06 00 00          ....4...
0DA0:0080 0C 20 61 3A 78 78 78 20-62 3A 79 79 79 0D 78 78  . a:xxx b:yyy.xx
0DA0:0090 20 62 3A 79 79 79 0D 00-00 00 00 00 00 00 00 00   b:yyy..........
0DA0:00A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0DA0:00B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0DA0:00C0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0DA0:00D0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0DA0:00E0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0DA0:00F0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
```

A 89 címen lévő 0D a parancssort lezáró kocsivissza, az ezt követő betűk pedig mutatják, hogy a DEBUG betöltésekor a parancssor hosszabb volt.

15.2.2 A PSP egy program indításakor

A programok indításakor a DOS a következő feladatokat látja el:

1. PSP létrehozása, memóriafoglalás, a PSP 5C-től és 6C-től kezdődően egy-egy FCB kitöltése, valamint a program indítási paramétereinek tárolása a PSP + 81-től kezdődően. A DOS a programnevet követő space-től a soremelésig található karaktersorozatot (string-et) indítási paraméterként kezeli és a jelzett helyre másolja. Nem kerülnek azonban ide a parancssorból a <, >, > >, valamint P| jelekkel kezdődő szavak, hanem ezeknek megfelelően a parancsfeldolgozó elvégzi az input, illetőleg output átrendeléseket, vagy gondoskodik a parancsok láncolásáról.

2. A programkód elhelyezése a PSP utáni területre és EXE programok esetén a relokáció elvégzése.

3. A DS, ES, SS és SP regiszterek beállítása. (A DS és ES mind a COM, mind az EXE kiterjesztésű programoknál a programszegmens elejére mutat. Az SS és SP regiszterek értékei EXE programoknál az EXE-fejléc szerint alakulnak, míg COM programoknál SS = DS és SP = 0FFFE).

4. A vezérlés átadása a kijelölt programkezdetre, ami COM programok esetén CS = DS:100, azaz a PSP-t követő program fizikai kezdete. EXE programok esetében a CS és az IP értéke az EXE-fejléc alapján állítódik be. Emlékeztetőül:

SS = Stack Segment regiszter
SP = Stack Pointer, azaz a veremtár címe SS:SP
DS = Data Segment
CS = Code Segment - ez jelzi az utasítások címének a szegmensét
IP = Instruction Pointer - ez mutatja a következő utasítás címének a helyét a szegmensen belül (cím offset-je), azaz a következő utasítás címe: CS:IP.

15.3 A programkörnyezet és a memórianyilvántartás

A fejezet eddigi részében megismerhettük a programszegmens felépítését, a program betöltésének és indításának folyamatát. Az ott leírtak szerves folytatását képezi a következő alfejezet, ugyanis az előzőekben csak érintőlegesen említett fogalmak részletezésére kerül itt sor. A PSP-ben található mutató adja meg mind a programkörnyezet, mind pedig a memórianyilvántartó blokk helyét. Ezek a területek a felhasználók számára elsődlegesen érdektelenek, de bonyolultabb struktúrájú programok létrehozásakor ezek ismerete hasznos lehet. Az, hogy ismeretük elsődlegesen érdektelen, azt jelenti, hogy általános igényű programokat írva, illetőleg ilyeneket használva nem fogunk olyan problémákkal találkozni, amelyek megértéséhez, elkerüléséhez vagy megoldásához foglalkozni kellene a memórianyilvántartással vagy a programkörnyezettel. Speciális igények esetén azonban más a helyzet, amint azt példán is be fogjuk mutatni.

15.3.1 Programkörnyezet

A programkörnyezet egy maximum 32 Kbyte hosszú karaktersorozat, amely a következő formában tartalmazza a DOS állapotát, területeit jelző és a program számára információkat adó paramétereket:

KULCSSZÓ = paraméter 0... ...KULCSSZÓ = paraméter 0 0

Mind a kulcsszavak, mind pedig a paraméterek ASCII karakterekből állnak. A kulcsszóhoz tartozó paraméter végét egy nullát tartalmazó byte jelzi. A programkörnyezet végét az utolsó paramétert lezáró nullát követően egy további nullát tartalmazó byte mutatja.

A programkörnyezet mindig tartalmazza a következő kulcsszavakat:

COMSPEC	- a hozzá tartozó paraméterérték egy elérési utat tartalmaz, amely kijelöli az aktuális COMMAND.COM helyét. Felhasználható például a COMMAND.COM programból való elindítására.
PATH	- a hozzá tartozó paraméterérték elérési utat (ill. pontosvesszővel elválasztva elérési utakat) tartalmazza. Ezt a PATH parancs állítja be és a parancsfeldolgozó számára jelöli ki az elérhető tartalomjegyzéket, illetőleg tartalomjegyzékeket. Részletesebb információk a PATH parancs leírásánál találhatók.

PROMPT - a hozzá tartozó karaktersorozat vezérli a parancsfeldolgozó kijelentkezésének formáját. A karaktersorozat megváltoztatását, illetőleg lecserélését a PROMPT parancs végzi. (Részletesebb információk a PROMPT parancs leírásánál találhatók.)

A fentieken kívül tetszőleges kulcsszó és paraméter bevihető a programkörnyezetbe a SET parancs segítségével. Ezt felhasználva olyan felhasználói program írható, amely a programkörnyezetben szereplő speciális kulcsszavakat fel tudja használni. Például egy grafikusan nyomtató program különféle típusú nyomtatókat ismer. Amennyiben a programkörnyezet nem tartalmazza a PRINTER= kulcsszót és ezt követően egy a program által ismert típusjelzést, úgy a rendszer megkérdezi a printer típusát. Ellenkező esetben a programkörnyezetben megadott típusú printert fog feltételezni. Ez a megoldás kényelmes a felhasználó számára, hiszen csak egyszer kell az AUTOEXEC.BAT file-ba beírnia egy SET PRINTER=típus parancsot, és ettől kezdve a rendszertöltés automatikusan közli a printer típusát az említett programmal.

Az aktuális programkörnyezet teljes tartalma az operációs rendszerből a paraméter nélkül megadott SET paranccsal kérdezhető le. Maximális mérete 32 Kbyte lehet. Vigyázni kell azonban, ha a tárban rezidens program található. Ekkor ugyanis a programkörnyezet nem haladhatja meg a rezidens program betöltésekor meglévő méretét. A méret túllépésének megkísérlése "Out of environment space" hibaüzenetet eredményez. Azaz a fenti üzenetet kapjuk, ha rezidens program mellett a PROMPT, a PATH vagy a SET paranccsal a korábbinál hosszabb paramétert, illetőleg a SET segítségével új kulcsszót akarunk megadni.

Példa:

Egy PATH \ parancs kiadása után, ha PRINT paranccsal kinyomtatunk egy file-t, akkor a PRINT már benne marad a memóriában újabb rendszertöltésig.

Ha ekkor kiadunk egy PATH A:\ parancsot, ami az "A" egység főkönyvtárát jelölné ki, ezt már a gép nem tudja elfogadni, a fenti hibaüzenetet kapjuk.

15.3.2 A memórianyilvántartás

A DOS 4B funkciója, amely betölti és elindítja a felhasználói programokat, a betöltés során a program rendelkezésére bocsájtja a lehetséges maximális memóriát. Ezért, ha egy program ilyen módon akar további programot indítani (overlay), gondoskodnia kell arról, hogy az általa nem használt területből felszabadítson annyit, amennyi az új program betöltéséhez elég. Ha az így betöltött és elindított program befejezte futását, akkor a hívó program újból lefoglalhatja saját számára ezt a területet. A memóriakezelés céljára három funkcióhívás szolgál:

A 48-as hívással tetszőleges méretű memóriablokkot lehet lefoglalni.
A 49-es hívással egy lefoglalt blokk felszabadítható.
A 4A-s hívással egy memóriablokk mérete módosítható.

Memóriablokk alatt az operációs rendszer egy egybefüggő memóriaterületet ért, melynek a helyét és méretét a rendszer tartja nyilván. Az indításkori memóriablokk méretét közvetve meg lehet kapni, mivel a PSP 2. byte-ján kezdődő szó megadja a legmagasabb szegmensként címezhető területet. (Amint a címzéseknél ismertettük, a memóriában minden cím egy szegmens és egy eltolási érték összegeként áll elő, ahol a szegmens 16 byte-os egységekben, azaz paragrafusokban van. Ez azt jelenti, hogy a cím a (szegmens*16) + eltolás képlet alapján számítódik.)

Amint azt a funkciók leírásánál ismertetjük is, a 48-as funkció, ha akkora memóriát kérünk, amennyi már nincs, visszaadja paragrafusban számítva a legnagyobb rendelkezésre álló memóriablokk méretét. Az is egy megoldás, hogy eleve a program mérete szerint módosítjuk a memóriablokk méretét.

A PC DOS Technical Reference nem közli a memóriakezelés minden részletét, de más forrásokból és ilyen problémákat megoldó programokból (pl. DEBUG, BASICA) a következők állapíthatók meg:

A DOS betöltő, amikor létrehoz egy memóriablokkot, a megelőző paragrafuson (eggyel kisebb szegmensen) egy memória leíró blokkot (Memory Control Block = MCB) hoz létre. Az MCB első byte-ja jelző. Ennek tartalma 4D, ha további memóriablokkok vannak még. Az utolsó memóriablokkot 5A jelzi. Ezt két szó követi. Az első megadja az MCB által foglalt memória kezdetének szegmenscímét (paragrafus). A második a kezelt memóriablokk méretét adja paragrafusban. Utolsó MCB esetén ez az érték a maradék szabad memóriaterület nagyságát adja. Nem utolsó memóriablokk esetén a következő memóriablokk szegmenscímét megkapjuk, ha a két szó tartalmát összeadjuk és az eredményt eggyel növeljük.

Táblázatosan az MCB felépítése:

1. byte	Jelző, tartalma 4D = nem utolsó blokk 5A = utolsó blokk
2-3. byte	Memóriablokk kezdetére mutató érték
4-5. byte	A nyilvántartott blokk mérete paragrafusban.

A fentiek ismeretében felvetődik a kérdés: hogy viszonyulnak a fenti funkcióhívások és az MCB egymáshoz?

A helyzet a következő: Ha 48-as funkcióhívással létrehozunk, vagy 4A hívással méretében módosítunk egy memóriablokkot, akkor új MCB keletkezik, és a korábbiban a méret átíródik.

Hasonlóan, 49-es hívás esetén, a megszüntetett blokk MCB-je kiiktatódik a láncból. (Az eddigiekből látható ugyanis, hogy a memóriablokkok nyilvántartását az MCB-k sorozata végzi, ahol a megelőző implicite tartalmazza az őt követő címét.)

Amennyiben a 49-es vagy a 4A funkcióhívásban az ES nem MCB-re mutató címet tartalmaz, úgy a hívás 9-es hibát jelez. (Carry magas és AL = 9)

Összefoglalva:

1. Minden programszegmens méretét a PSP-t megelőző szegmensen található memória leíró blokk (MCB) tartja nyilván.

2. A DOS funkcióhívás memóriakezelése MCB-k tartalmának módosításával, illetőleg ezek létrehozásával valósul meg.

3. Egy memóriablokk - ez lehet programszegmens vagy önállóan létrehozott memóriablokk - mindig akkora méretű, amekkorának lefoglaltuk vagy módosítottuk.

4. Ha kezelni akarunk egy memóriablokkot, akkor mindig a blokk kezdetének szegmenscímével kell azonosítanunk. (Ezt az értéket kell az ES regiszterbe tenni a 49-es és a 4A funkcióhívás előtt, illetőleg ezt az értéket kapjuk vissza az AX regiszterben, ha a 48-as funkcióhívás eredményes volt.)

A memóriaallokáció első szegmensének (a lánc első elemének) címét a PSP 16-17. byte-jából olvashatjuk ki. Ezután a láncon végighaladva megtudhatjuk azt is, milyen tárrezidens programok kerülnek a memóriába.

5. Amennyiben programhiba miatt valamelyik MCB felülíródik, a program terminálásakor a DOS "Memory allocation error - System halted" hibaüzenetet ír ki. Újra kell tölteni a DOS-t (boot)!

Végül megjegyezzük, hogy ha egy program hibakezelése nem megfelelő, akkor előfordulhat, hogy a DOS-hoz kerül a vezérlés (pl. lemezműveleti hibára abort választ adva) anélkül, hogy a program területe felszabadulna. Kisebb memóriájú gépeknél, vagy ha ez egymás után többször ismétlődik, elfogyhat a szabad memória. Ilyenkor a rendszer egy újabb program indítására "No memory" hibaüzenetet ad.

Az eddigiekből látható, hogy a memóriakezelés meglehetősen nehézkes, ezért ilyenkor a legegyszerűbb megoldás a memória felszabadítására az új rendszertöltés. <Ctrl-Alt-Del>

15.4 A file-kezelés és munkaterületeinek szerkezete

A PC DOS operációs rendszer legkorábbi változata egyetlen, a CP/M operációs rendszerhez hasonló file-kezelési eljárást ismert. Ennek a lényege, hogy minden file-művelethez egy, a művelet minden fontos paraméterét tartalmazó, meghatározott szerkezetű terület, az úgynevezett file leíró blokk (FCB), valamint egy, az operációs rendszer által nyilvántartott munkaterület, a lemez átviteli terület (DTA) szükséges. Ezeket a felhasználó a memória tetszőleges helyén kijelölheti. Az operációs rendszer a file létrehozásához, megnyitásához és lezárásához csak a file leíró blokkot, olvasási és írási műveletekhez a lemez átviteli területet is igényli. Az írási és olvasási műveletek mindig az FCB-ben megadott méretű blokkokban történnek. A file munka közbeni állapotáról információkat a felhasználó a file leíró blokkból szerezhet. A továbbiakban az eljárást, illetőleg a hozzá tartozó funkcióhívásokat a file leíró blokk neve alapján FCB-orientált eljárásnak, illetőleg funkcióhívásoknak fogjuk nevezni.

A 2.00 és magasabb verziószámú DOS operációs rendszerek többek között egy új file-kezelési móddal is kiegészültek. Ennél a módszernél a felhasználó minden átviteli műveletnél megadja az átvitelre kerülő adatok helyét és mennyiségét, továbbá a file azonosítása nem a file leíró blokkal, hanem egy sorszámmal, az úgynevezett file-számmal történik. Ezt az operációs rendszer közli a felhasználóval a file megnyitásakor. A file-név megadása így csak a file megnyitásakor szükséges. Ezt az eljárást file-számos filekezelésnek fogjuk hívni.

A fejezet további részei először kiemelten ismertetik a file-kezelési eljárások fogalmait, majd külön szakasz foglalkozik az egyik és a másik eljárás használatával is. Bár ez a feldolgozás a legelső olvasáskor egy kicsit több munkát igényel az olvasótól, mert az egyes fogalmak részleteihez vissza kell lapoznia, viszont a későbbiekben, amikor csak egy-egy részletre van szükség, lényegesen kényelmesebbé teszi a könyv használatát.

15.4.1 File leíró blokk (FCB)

A file leíró blokk (File Control Blokk = FCB) az a terület, ahol az FCB-orientált filekezelés során a file valamennyi fontos információja megtalálható. Megkülönböztetünk normál és bővített FCB-t. Egy FCB lehet továbbá megnyitott és meg nem nyitott.

A normál file leíró blokk szerkezete:

Cím byte (decimálisan)	**Tartalom**
0.	Lemezegység azonosító, értéke: megnyitás előtt: 0 - aktuális egység 1 - A egység 2 - B egység stb. megnyitás után: 1 - A egység 2 - B egység stb. Ha megnyitás előtt az értéke 0, akkor megnyitó (0F) DOS funkció az aktuális egységnek megfelelően átállítja. Ha tartalmának megfelelő sorszámú egység nincs vagy hibás, akkor a funkció végrehajtása sikertelen lesz, és hibajelzést kapunk.
1-8.	File-név (kiterjesztés nélkül). A mezőben a file-nevet balra igazítva kell elhelyezni, és a 8 karakternél rövidebb nevek végét space-szel kell feltölteni.
9-11.	Kiterjesztés. A file-név kiterjesztését kell megadni benne, pont nélkül.
12-13.	Aktuális blokk sorszáma. Azt mutatja, hogy a file kezdetéhez viszonyítva hányadik 128 byte-ból álló blokk feldolgozása következik. A rekord méretét a logikai rekordhossz mező határozza meg. Az aktuális blokk sorszáma mezőt a file megnyitása (0F funkció) 0-ra állítja és a szekvenciális írás és olvasás funkcióhívások (14-es és 15-ös funkció) használják. (Egy blokk 128 db rekordból áll. Ezek hosszát a következő mező mutatja.)
14-15.	Logikai rekordméret (szó). Byte-okban megadja egy logikai rekord hosszát. A file megnyitásakor a rendszer az értékét 128-ra, azaz hexadecimális 80-ra állítja. Amennyiben ettől eltérő rekordmérettel akarunk dolgozni, úgy a megnyitás után át kell írni.

16-19. File-méret (duplaszó). A file byte-okban mért teljes hosszát tartalmazza. A méret a duplaszó szervezésnek megfelelően fordított sorrendben tartalmazza a számot.

20-21. Dátum (szó). A file létrehozásának, illetőleg utolsó módosításának dátumát tartalmazza év/hó/nap formában a következő módon:

```
| 21. byte      |    20. byte     |
F E D C B A 9 8 7 6 5 4 3 2 1 0  bitek
|      év     | hónap |   nap     |
```

22-31. Foglalt a rendszer számára.

32. Aktuális rekord sorszáma a blokkban. Értéke 0-127 között lehet, és megadja, hogy az aktuális blokkon belül hányadik rekord kerül feldolgozásra. Ezt a mezőt a 0F funkció nem inicializálja, ezért beállításáról a felhasználónak kell gondoskodnia.

33-36. Rekord abszolút sorszáma (duplaszó). A file elejétől számítva adja meg az aktuális rekord sorszámát. Az első rekord sorszáma 0. A sorszám értékét a random írás és random olvasás (lásd 28-as és 27-es funkció leírását) használja, és beállításáról a felhasználónak kell gondoskodnia.

A nem megnyitott FCB-ben csak a 0-11 byte-okat kell kitölteni. A további mezőket a file megnyitási, illetőleg -létrehozási funkcióhívások során a rendszer tölti ki.

Meg nem nyitott FCB-re találhatunk példát a PSP felépítését ismertető részben. Megjegyezzük, hogy ha a PSP-ben 5C címen kezdődő, az operációs rendszer által létrehozott FCB-t használjuk, figyelni kell arra, hogy annak utolsó byte-ja a 80-as címre esik. A 80-as címtől kezdődik a PSP-ben a DTA terület, ezért, ha ezt nem állítjuk át, az FCB vége átíródhat.

A normál FCB segítségével csak a normál attributumú file-ok kezelhetők. Ha a felhasználó rejtett, rendszer vagy csak olvasható file-okat akar kezelni, akkor bővített FCB-t kell használnia.

A bővített FCB egy, az elején egy 7 byte-os prefixet tartalmazó file leíró blokk.

A bővített file-leíró blokk szerkezete:

Cím
byte
(dec.) Tartalom

0. Jelző, értéke FF kell legyen. Ez jelzi a rendszer számára, hogy bővített FCB következik.

1-5. A rendszer számára foglalt.

6. File attributum. Azok a DOS funkciók, amelyek hívásakor FCB-t kell megadni, a megadott FCB-ben kijelölt attributumú (tulajdonságú, pl. rejtett) file-okkal fognak foglalkozni.

Az attributum byte bitjeinek jelentése:

6-7. bit értéke 0.

5. bit Archív bit. File-ba írás után értéke 0-ra vált, és a sikeres file-lezárás (10-es funkció) állítja 1-re (hexadecimális 20 a byte értéke normál és jól lezárt file-nál).

4. bit Altartalomjegyzék jelző. Ez a bejegyzés altartalomjegyzék nevet jelöl ki.

3. bit Kötetnév jelző. Ez a bejegyzés az adathordozó nevét jelöli. A byte többi bitje 0 kell legyen, azaz az attributum byte értéke ekkor 8.

2. bit Rendszer-file. A file csak speciális módon érhető el. Normál DOS műveletekben nem vesz részt.

1. bit Nem látható (rejtett, hidden) file. A file a DIR listán nem jelenik meg. Ezt az attributumot a file létrehozásakor, kreálásakor kell megadni.

0. bit Csak olvasható file. A file a PC DOS-szal közvetlenül nem törölhető, nem írható felül.
A 0-2. bitek tetszőleges értékkombinációja előfordulhat. Leggyakoribbak:
20 - normál file,
27 - csak olvasható, rejtett rendszer file,
21 - csak olvasható file.

7-41 Normál FCB. Az itt elhelyezkedő FCB teljes mértékben megegyezik az előzőekben ismertetekkel.

15.4.2 Lemezátviteli terület (DTA)

A lemezátviteli terület (Disk Transfer Area = DTA) az FCB-orientált file-kezeléskor az I/O puffer szerepét játsza, azaz a memóriának az a része, ahova az olvasási műveletek az adatokat a file-ból beírják, és ahonnan az írási műveletek az adatokat veszik.

A DTA méretét mindig a file leíró blokk decimális 14-15 byte-ján található érték mutatja. Ha ezt az értéket nem írjuk felül, akkor a DTA mérete 128, azaz hexadecimális 80 byte. (Ezek a file-műveletek mindig egy logikai blokkot mozgatnak a DTA-ról, illetőleg a DTA-ra.)

A DTA helyét a felhasználó az 1A funkció hívásával állíthatja be. Amennyiben nem állítjuk be, úgy a rendszer a DTA-t a PSP 80. byte-jától kezdi.

15.4.3 File-kezelés az FCB segítségével

Az FCB-t felhasználó DOS funkcióhívások a következő műveleteket végzik el:

file-műveletek:
- file megnyitása
- file létrehozása
- file lezárása

- szekvenciális (soros) írás
- szekvenciális (soros) olvasás
- közvetlen elérésű (random) írás
- közvetlen elérésű (random) olvasás

tartalomjegyzék kezelés:
- első bejegyzés keresése
- további, a megtalálttal egyező bejegyzés keresése
- file törlése
- file átnevezése

Ehhez a műveletcsoporthoz tartozó kiegészítő funkció a DTA beállítása.

A fenti funkciókat részletesen a DOS funkciókat leíró fejezetben ismertetjük majd.

Mint a fejezet bevezető részében leírtuk, ennél a file-kezelési módnál a file neve, a file-t tartalmazó adathordozó helye az FCB megfelelő mezőjében előzetesen beállítandó. Vagyis első lépésként mindig létre kell hozni egy megfelelő FCB-t.

Ezután kiadható egy file megnyitó vagy file létrehozó funkcióhívás. (Csak már létező file-t lehet megnyitni, de output esetén újabb rekordokkal kiegészíthető az így megnyitott file. Ezzel szemben tetszőleges nevű file-t létrehozhatunk, de ha létezett már ilyen nevű file, akkor annak korábbi tartalma elvész.)

A file-ba való írás, illetőleg a file-ból való olvasás előtt be kell állítani a DTA címét, vagyis a memóriában lévő munkaterületünk címét.

Közvetlen írási vagy olvasási műveleteknél be kell még állítani (ha szükséges) az elérni kívánt rekord vagy blokk mutatóját.

További kötöttsége ennek a file-kezelésnek, hogy a DTA mérete maximum 64 Kbyte lehet, de semmiképp sem lépheti túl a kijelölt szegmens felső határát. Ez utóbbi megkötés azt jelenti, hogy például 2000:F000 DTA kijelölése esetén a DTA maximum 2000:FFFF-ig terjedhet, azaz maximum hexadecimális 1000 byte-os blokk olvasható vagy írható.

Ezzel a funkciócsoporttal egy file szekvenciális beolvasásának menete a következő:

1. Kialakítunk egy meg nem nyitott FCB-t. Kezdődjön ez az INPUTFCB nevű tárterületen, és tartalmazza a "PROBA.KIT" nevet, továbbá jelezzük, hogy az aktuális egységről akarunk beolvasni. Ezt Microsoft MASM makró-assembler számára a következőképpen írhatjuk le. (A továbbiakban is a példákat így, MASM-ban adjuk meg.)

```
INPUTFCB DB 0               ; egységazonosító
         DB "PROBA   "      ; file-név
         DB "KIT"           ; kiterjesztés
         DB 24 DUP(0)       ; 24 byte FCB részére
```

Legyen ez a terület a kódszegmensen elhelyezve. Ekkor a

```
PUSH CS
POP  DS
```

utasításpárral beállíthatjuk a megfelelő adatszegmens értéket. A továbbiakban feltételezzük, hogy az adatszegmens értéke mindig jól be van állítva.

2. A file megnyitása inputra, mely a következő utasításokkal érhető el:

```
MOV  DX,OFFSET INPUTFCB          ; DS:DX beállítása FCB címére
MOV  AH,0FH                      ; a funkció száma
INT  21H

                                 ; az AH-ban megadott funkció
                                 ; hívása (a DOS funkciók a
                                 ; 21-es megszakításon keresztül
                                 ; érhetők el).
```

Igényes program a funkcióhívás után ellenőrzi, hogy az sikeres volt-e. Itt most ettől eltekintünk.

```
MOV CX,0                         ; file méret
```

3. DTA beállítása. Használjuk erre a célra a program legvégén elhelyezkedő "MUNKA" nevű területet. A DTA beállításához szükséges utasítások:

```
MOV DX,OFFSET MUNKA              ; DS:DX beállítása
ADD DX,CX
MOV AH,1AH                       ; funkciószám
INT 21H                          ; funkcióhívás
```

4. 1 rekord beolvasása, feltételezve, hogy rekordonként akarunk olvasni.

```
MOV DX,OFFSET INPUTFCB           ; DS:DX beállítása FCB címére
MOV AX,14H                       ; funkciószám
INT 21H                          ; funkcióhívás
```

5. File vége vizsgálat

```
CMP AL,0                         ; összehasonlítás
JZ  VAN_MEG                      ; nincs file vége, ennek
                                 ; megfelelő címkére lép
CMP AL,1                         ; összehasonlítás
JZ  VEGE1                        ; vége, elugrás
CMP AL,3                         ; hasonlítás
JZ  VEGE2                        ; vége, elugrás
...
```

A VEGE1 és VEGE2 abban különbözik, hogy a VEGE2 esetén töredék blokk került beolvasásra és ennek méretét hozzá kell adni az eddig számolt mérethez, amelyet jelenleg a CX regiszter tartalmaz.

6. Ha nem volt file vége, akkor a DTA-t, továbbá a file méretét növelni kell.

```
ADD  CX,128                      ; file-méret növelés
```

A 3-6. lépést a file végéig kell ismételni.

7. File lezárása.

```
MOV DX,OFFSET INPUTFCB            ; DS:DX beállítása az FCB címére
MOV AH,10H                        ; funkciószám
INT 21H                           ; CLOSE funkcióhívás
```

Megjegyezzük, hogy ez a fajta - 128 byte-os rekordokban való - olvasás lassú. Ezért nagyobb file-okat célszerűbb blokkonként írni/olvasni. Ezt az eljárást egy másik példán mutatjuk be.

Írjuk ki az előző példában beolvasott file-t az "A" egységre, "UJ"-ra módosított kiterjesztésű névvel. Ezt a feladatot a következő programmal valósíthatjuk meg:

1. lépés: FCB módosítása

```
MOV BYTE PTR [INPUTFCB],1           ; "A" egység
MOV BYTE PTR [INPUTFCB + 9],"U"     ; kiterjesztés
MOV BYTE PTR [INPUTFCB + 10],"J"    ; átírása
MOV BYTE PTR [INPUTFCB + 11]," "    ; átírása
```

2. lépés: File létrehozása

```
MOV DX,OFFSET INPUTFCB              ; DS:DX beállítása FCB címére
MOV AH,16H                          ; file létrehozás
INT 21H                             ; funkcióhívás
```

3. lépés: Megnyitott FCB beállítása. Az FCB 32. byte-ját 0-ra kell állítani.

```
MOV BYTE PTR [INPUTFCB + 32],0
```

4. lépés: DTA beállítása

```
MOV DX,OFFSET MUNKA                 ; DS:DX beállítása
MOV AH,1AH                          ; funkciószám
INT 21H                             ; funkcióhívás
```

5. lépés: Kiírandó blokkok számának kiszámítása.

```
PUSH CX                             ; file-méret mentése CX-be
MOV AX,CX
MOV CL,128                          ; blokkméret
DIV CL
MOV CL,AL                           ; eredmény CX-be
MOV CH,0
INC CX                              ; töredék blokk miatt
```

A 28-as funkcióhívás a CX-ben beállított darabszámú 128-byte-os blokkot ír ki folytonosan a DTA-ról, az FCB szerint.

6. lépés: Blokk közvetlen írása

```
MOV DX,OFFSET INPUTFCB          ; DS:DX beállítása
MOV AH,28H                      ; funkciószám
INT 21H                         ; funkcióhívás
```

7. lépés: File-méret beállítása az FCB-ben

Erre azért van szükség, mert a töredék blokk helyett teljes 128 byte-os blokkot írtunk ki. Ezt úgy is korrigálhatjuk, hogy a file lezárása előtt az FCB-be beírjuk a tényleges méretet.

```
POP CX                          ; méret visszahozása
AND CL,7FH
MOV [INPUTFCB + 16],CL          ; elég a méret végét átírni
```

8. lépés: File lezárása

```
MOV DX,OFFSET INPUTFCB          ; DS:DX beállítása FCB címére
MOV AH,10H                      ; file-zárási funkció beállítása
INT 21H                         ; funkcióhívás
```

Látható, hogy több rekordot együtt írva lényegesen egyszerűbb a file kiírása, de kihasználtuk azt, hogy 64 Kbyte-nál kisebb file-ról van szó. Ennél nagyobb file esetén blokkos adatmozgatás használata esetén is ciklust kell szervezni.

Nem ilyen egyszerű a file olvasási-, illetőleg írási feladat, ha tetszőleges tartalomjegyzékben elhelyezkedő file-t kell mozgatni. Ekkor a file megnyitása előtt a 3B funkcióhívással ki kell jelölni a kívánt egységen a megfelelő tartalomjegyzéket aktuális tartalomjegyzéknek. Ha a file nevét például konzolról kérdezzük meg, akkor - az FCB felépítése miatt - kényelmetlen string átalakításra van szükség. Ugyanezek a megjegyzések igazak, ha nem írni vagy olvasni akarunk, hanem file-t keresünk, törlünk, vagy átnevezünk.

15.4.4 A DOS file-szám (handle) és használata

Láttuk, hogy az FCB-s file-kezelés elég sok kötöttséget jelent. Különösen érezhető ez, ha a különböző altartalomjegyzékekkel dolgozunk. Vigyázni kell arra is, hogy a feldolgozás kezdetére az FCB helyesen legyen kitöltve, és a feldolgozás során ne romoljon el. Lényegesen kényelmesebb munkát biztosít a file-számos file-kezelés. Ilyenkor a megnyitáshoz, törléshez elég a DOS megszokott formátumában megadni a file nevét. Anélkül, hogy az előző részhez hasonlóan itt is részleteznénk, nézzük végig itt is egy file beolvasásának vagy írásának vázlatos menetét.

1. File megnyitása. A megnyitáshoz ki kell alakítani egy karaktersorozatot, amely folyamatosan tartalmazza a file-t tartalmazó tartalomjegyzék elérési útját és a file nevét. A string végét egy nulla byte jelzi. (Ha csak file-nevet adunk meg, akkor az aktuális egység aktuális altartalomjegyzékét feltételezi a rendszer.)

Amennyiben file-névként a rendszer szabványos periférianeveit vagy a CONFIG.SYS-ben DEVICE által kijelölt periférianevet adunk meg, úgy a rendszer ezt a perifériát fogja file-ként kezelni.

A megnyitáshoz a DS:DX mutatja a file-nevet tartalmazó string kezdetét, AL pedig a file elérési módot (0 = olvasás, 1 = írás, 2 = írás-olvasás) jelzi. A megnyitás a 3D funkcióhívással (INT 21, AH = 3D) történik. Sikeres megnyitás esetén a funkció AX-ben egy 4-nél nagyobb értéket ad vissza. Ez az érték a file-szám. A továbbiakban minden file-műveletnél a BX-nek ezt a számot kell tartalmaznia, mert ez mutatja a rendszernek, hogy melyik file-ra vonatkozik az utasítás. Ha az AX 0-4 közötti értéket tartalmaz, akkor a DOS standard input/output perifériáit akarjuk kezelni. (Ezeket a rendszer a program indítása előtt automatikusan megnyitja.) (Részletesebb információ a 3D funkcióhívás leírásánál található.)

2. Ha szekvenciálisan akarunk írni vagy olvasni, akkor itt következhet az író-, illetőleg olvasó utasítást tartalmazó ciklus.

Itt nincs megkülönböztetve szekvenciális, random rekord, valamint random blokk olvasás/írás.

Egyetlen író, illetőleg olvasó utasítás van. A mozgatandó byte-ok számát a CX regiszter tartalmazza. Az adatok memóriabeli kezdetét a DS:DX jelöli ki. (Itt nem okoz problémát, ha DX + CX > 64 Kbyte, azaz az adatmozgatás szegmenshatárt lép át, a rendszer lekezeli ezt az esetet is.)

A file-ban az adatmozgatás kezdetét egy, a file-hoz rendelt mutató, az úgynevezett file-pointer tartalmazza. A file-pointer egy 4 byte-os szám, amely tartalmazza, hogy a file elejétől számítva hányadik byte-tól akarunk írni, illetőleg olvasni.

A file megnyitás, illetőleg létrehozás a file-pointert 0-ra állítja, az írás és olvasás pedig a mozgatott terület végére.

A fentiekből következik, hogy szekvenciális file-kezelésnél a file-pointerrel nem kell foglalkozni.

3. Random file-kezelésnél szükség van a file-pointer kezelésére.

Erre a célra a 42-es funkcióhívás szolgál, amellyel a file kezdetétől, végétől vagy pedig aktuális értékétől számítva tetszőlegesen átállíthatjuk a file-pointert.

4. A munka végén 3E funkcióhívással le kell zárni a megnyitott file-okat. Ha ezt elmulasztjuk, output file-ok esetén a teljes file elvész.

Megjegyzés:

1. A 45-ös funkcióhívással mód van file-kezelő duplikálásra is. Ez azt jelenti, hogy ugyanarra a file-ra egyidejűleg két különböző file-számmal hivatkozhatunk.
2. Elsősorban nem igazi file-ok, hanem perifériákra mutató file-számok esetén van jelentősége, hogy a 44-es funkcióhívással lekérdezhetjük, illetőleg beállíthatjuk a file-ra vonatkozó vezérlő információkat. Például a funkcióhívással ellenőrízheti a program, hogy a feltételezett felhasználói perifériakezelő ténylegesen installálva van-e a rendszerben, vagy helyette egy valódi lemez-file-t nyitott-e meg a file-megnyitási utasítás.

16. DOS megszakítások

A DOS lehetőséget ad arra, hogy felhasználója hozzáférhessen a software megszakításokhoz (software interrupt-okhoz). Ezek használatán keresztül a rendszerprogramozó jobban megismerheti az operációs rendszer belső működését is. A fejezet tartalmának megértéséhez szükséges alapvető ismeretek (8086 processzor regiszterei, megszakítási rendszere, interrupt vektortábla felépítése, a példákban használt írásmód) megtalálható a DOS struktúráját ismertető korábbi fejezetben.

A PC DOS a megszakítások közül a hexadecimális 20-adiktól a 3F sorszámúig terjedőket használja. Az ezekhez tartozó megszakítási vektorok a 0. memóriaszegmens hexadecimális 80-as címtől az FF címig találhatók. A továbbiakban, ha nem jelezzük külön, akkor szám alatt (így a megszakítások és funkcióhívások sorszámai alatt is) hexadecimális számot értünk - bár ezt nem fogjuk mindig jelezni. Az egyes megszakításokat jelöljük majd így is:

INT *megszakítás sorszáma* pl.: INT 21

16.1 A 20-as megszakítás: program befejezése

Ezzel a megszakítással fejezhetik be a programok a futásukat és adhatják vissza a vezérlést a PC DOS-nak. Meghívásakor a rutin visszaállítja a felhasználói program indítása előtti Ctrl-Break-, kilépési- és kritikus hibákat kezelő vektor címet (INT 23, INT 22, INT 24). Valamennyi file-puffer tartalma törlődik. Használatakor, meghívása előtt, a következőkre kell figyelemmel lenni:

- minden, a felhasználói program által módosított file-t le kell zárni (lásd a 10., illetve 3E. funkcióhívást), különben a tartalomjegyzékben a file-méret, felírási idő, felírási dátum információk helytelenek lehetnek,
- gondoskodni kell róla, hogy a CS regiszter által kijelölt szegmens a Program szegmens prefix (PSP) vezérlőblokkra mutasson.

Mivel a DOS 4C és a 31 funkcióhívásaival a programból való visszatérés feladatát kényelmesebben meg lehet oldani - azokkal ugyanis visszatérési kódot is át tudunk adni -, célszerűbb azokat használni.

16.2 A 21-es megszakítás: DOS funkció meghívása

Részletes ismertetésére a 17. fejezetben kerül sor.

16.3 A 22-es megszakítás: program kilépési cím

A megszakítási vektortáblában a megszakításnak megfelelő helyen nem rutincím található, hanem az a cím, amelyre a DOS a program befejeződése után a vezérlést átadja. Új programszegmens létrehozásakor ezt a címet a DOS bemásolja a PSP-be. A megszakítás a felhasználók által nem hívható.

16.4 A 23-as megszakítás: Ctrl-Break kilépési cím

Az operációs rendszer ezt a megszakítást hívja meg, ha a < Ctrl-Break > leütése érvényre jut. Mint arról a "Rendszer konfigurálása" fejezetben szó volt, a < Ctrl-Break > billentyű felhasználó általi leütése csak akkor lesz hatásos, ha

BREAK = ON	állapot esetén tetszőleges,
BREAK = OFF	állapot esetén konzol, printer vagy aszinkron I/O műveletet megvalósító funkcióhívás kerül

végrehajtásra. (A 6-os és 7-es funkcióhívások végrehajtásakor lesz megszakítás.)

Amennyiben a felhasználó saját Ctrl-Break kilépési rutint ír és gondoskodik valamennyi regiszter lementéséről, úgy IRET (visszatérés interruptból) utasítással folytathatja a felhasználói programját.

A DOS Ctrl-Break megfelelő előkészítés után INT 20 segítségével visszalép a parancsfeldolgozóhoz. Ha a Ctrl-Break kilépési rutin használja a 9-es vagy az A-dik funkcióhívást, akkor a C-t a DOS kocsivissza, soremelés (CR,LF) karakterekre cseréli.

Ha IRET hatására a program tovább fut, akkor az aktuális I/O művelet új sor elején folytatódik.

A megszakítás érvényrejutásakor valamennyi regiszterben az az érték lesz, amelyet a regiszter az eredeti DOS funkcióhíváskor tartalmazott.

A felhasználói Ctrl-Break rutinra semmiféle megszorítás nincs, még a DOS funkciókat is szabadon használhatja. IRET kiadása előtt azonban valamennyi regiszterben a belépési értéknek kell lennie.

Ha a rutin a visszatérést távoli return (RET FAR) segítségével oldja meg, akkor a carry flag használható annak közlésére, hogy a program folytatódjék-e. (IRET esetén a CF értéke NO.)

Amennyiben egy program új programszegmenst nyit, és az ebben futó program kicseréli a Ctrl-Break megszakítás címét, úgy a második program befejezésekor, az első programba való visszalépés előtt, a DOS a második program PSP-jéből visszaírja az első program futása alatt érvényes Ctrl-Break címet.

A 22-es megszakításhoz hasonlóan, felhasználói programból ne hívjuk meg ezt az interruptot.

16.5 A 24-es megszakítás: kritikus hibák kezelése

A DOS akkor hívja meg ezt a megszakítást, ha kritikus hibát észlel.

Meghívásakor az AH regiszter tartalmazza a hiba típusát jellemző információkat. Az AH regiszter 7. bitje 0 lesz lemezhiba esetén. Karakteres berendezés vagy FAT hibájakor az AH további információkat csak lemezhiba esetén tartalmaz, ezért ezek ott kerülnek ismertetésre.

A BP:SI mutat az aktuális DHC blokk-ra. (Aktuális DHCB = Device Header Control Block = azon berendezés vezérlő információit tartalmazó blokk, amelynél a hiba fellépett.) A DHCB részletes ismertetése megtalálható "A PC DOS belső struktúrája" című 13. fejezet ezzel foglalkozó részében, továbbá a 44-es funkció leírásánál. Innen a hibára vonatkozó további információk nyerhetők.

A hibakód a DI regiszter alacsonyabb helyiértékű byte-jában található az alábbiak szerint:

Hibakód	**Leírás**
0	Olyan lemezre próbáltunk írni, amely írásvédett.
1	Ismeretlen egység.
2	A meghajtó nincs bekapcsolt állapotban.
3	Ismeretlen parancs.
4	Adathiba (CRC).
5	Hossz hiba.
6	Pozicionálási hiba.
7	Ismeretlen típusú adathordozó.
8	A szektor nem található.
9	A papír kifogyott a nyomtatóból.
A	Írási hiba.
B	Olvasási hiba.
C	Nem meghatározható hiba.

Mielőtt a PC DOS ezt a hibarutint meghívná, ötször próbálja meg a kijelölt műveletet elvégezni. Csak ha mind az öt kísérlet sikertelen volt, akkor kerül sor az INT 24 meghívására.

Az INT 24-be való belépés előtt az általános regisztereket a PC DOS úgy állítja be, hogy a hibás művelet azok alapján újra ismételhető legyen.

A felhasználó természetesen ezt a hibakezelő rutint is kiválthatja saját eljárással. Ahhoz, hogy a DOS hibakezelés a felhasználó hibakezelő rutinjához kerüljön, majd ezt követően a DOS-hoz térjen vissza, a felhasználónak, mielőtt a 24-es megszakítás bekövetkezhetne, el kell tennie a 24-es interrupt vektort, és ezt olyannal kell helyettesítenie, amely a felhasználói hibakezelő rutinra mutat.

A 24-es megszakítás bekövetkezése után, amikor rutinunk megkapta a vezérlést, a flag regisztert a verembe kell tenni (PUSHF), és meg kell hívni az eltett INT 24 interrupt vektort (CALL FAR). Ekkor a DOS hibaüzenetet ad ki és vár, hogy a felhasználó milyen választ ad (Abort, retry, ignore). Ezután a DOS visszaadja rutinunknak a vezérlést a CALL FAR utáni utasításra.

A felhasználói verem (stack), amely a következő felépítésű (a verem tetejétől számítva), felhasználható arra, hogy a hiba lekezelése után ne térjünk vissza a DOS-hoz.

FLAG-ek
CS
IP visszatérés a 24-es megszakításból a felhasználói programhoz
ES

DS
BP
DI
SI Felhasználói regiszterek az eredeti 21-es funkcióhívás előtt
DX
CX
BX
AX
FLAG-ek
CS Visszatérés a hibakezelésből abba a DOS funkcióba, ahol a hiba fellépett
IP

Ilyen közvetlen visszatérésnél a verem tetejétől számított decimális 13., 14., 15. byte-on található FLAG, CS, IP-re kell az IRET utasítást kiadni (pl. úgy, hogy az IRET előtt D-vel megnöveljük az SP értékét). Arról a felhasználónak kell gondoskodnia, hogy a visszatérés utáni első funkcióhívásnak 12-nél magasabbnak kell lennie, különben a PC DOS meghatározatlan állapotba kerül.

Az INT 24-et lezáró IRET utasítás végrehajtása után a DOS az AL regiszter tartalmának megfelelően a következőképpen jár el:

(AL) = 0 A hiba figyelmen kívül marad.
= 1 A művelet ismétlése.
= 2 A program a 23-as megszakítással befejeződik.
= 2 Rendszerhívási hiba.

A felhasználó által írt hibakezelő rutinnak a következő regisztereket mentenie kell: SS, SP, DS, ES, BX, CX, DX, továbbá csak a 12-nél alacsonyabb rendszerfunkciókat használhatja. Ügyelni kell arra a felhasználó által írt hibakezelő rutin írásakor, hogy az INT 24-et a DOS mindig interrupt - letiltott - állapotban adja ki.

Lemezhiba

Lemezhibák esetén ez a rutin csak akkor aktiválódik, ha a hiba DOS funkció végrehajtása során fordult elő. (Pl. INT 21, AH = 14, szekvenciális olvasás.) Nem kerül meghívásra az INT 24, ha a lemezhiba INT 25 vagy INT 26 során lép fel.

Ha lemezhiba fordul elő (az AH 7. bit-je 0), akkor az AL regiszter a hibás meghajtó azonosítóját tartalmazza. (A bitek az alacsonyabb helyiértéktől számítva: 0 = A:meghajtó; 1 = B:meghajtó stb.) Az AH 0-2 bitjei pedig a következő információt tartalmazzák:

0. bit = 0 a hiba olvasási műveletnél
= 1 a hiba írási műveletnél történt
2.- 1. bit a lemez terület típusára utal:

0	0	DOS terület - rendszer állományok
0	1	FAT terület
1	0	tartalomjegyzék
1	1	adatterület

Ne feledjük, hogy a hibakód ilyen esetben is a DI regiszter alacsony helyiértékű byte-ján található. (Lásd az INT 24-ről általánosan leírtakat.)

Az AH regiszter 3-5. bitjei mutatják, hogy a hiba kezelésére milyen válasz adható.

Bit	**Jelentés**
3	=0 FAIL nem megengedett =1 FAIL megengedett
4	=0 RETRY nem megengedett =1 RETRY megengedett
5	=0 IGNORE nem megengedett =1 IGNORE megengedett

Ilyenkor értelemszerűen a BP:SI nem tartalmaz használható információt.

Egyéb hibák

Ha az AX regiszter 7. bitje 1, akkor a hiba karakteres berendezésen lépett fel, vagy az is előfordulhat, hogy a FAT bejegyzés hibás. Ekkor a DHC blokkból eldönthető, hogy melyik eset fordult elő: Ha az attributum byte magas helyiértékű bitjei blokk-orientált egységre utalnak, akkor a FAT hibás. Különben a hiba karakteres berendezésen fordult elő.

Ha karakteres berendezésről van szó, akkor az AL tartalma semmitmondó. Mint az INT 24 elején írtuk, a BP:SI mutat arra a DHCB-re, amelynél a hiba felmerült. Ez a következőket tartalmazza:

Cím	**Adat**	**Az adat értelmezése**
(hányadik	típus	byte)
0-3.	duplaszó	A következő DHC blokk kezdő címe. Ha értéke FFFF, akkor ez az utolsó létező karakteres berendezés DHCB-je.
4-5.	szó	A berendezés típusának leírása, az alábbi bit kiosztás szerint: 15. bit = 0, akkor blokkos berendezés = 1, akkor karakteres berendezés 14. bit tartalmazza az Input-Output Controll (IOCTL) bitet 3. bit = 1, ha az aktuális CLOCK- (óra-) egységről van szó 2. bit = 1, ha az aktuális NUL-egységről van szó 1. bit = 1, ha az aktuális standard output egységről van szó 0. bit = 1, ha az aktuális standard input egységről van szó. (Értelemszerűen a 0.-3. bitek közül maximum az egyik lehet egyenlő eggyel. A fel nem sorolt bitek tartalma nulla.)
6-7.	szó	A handler által feldolgozandó parancs-blokk értelmezését végző rutin címe.
8-9.	szó	Az aktuális berendezés parancsfeldolgozó rutinjának címe.
A-11.	8 byte	Név mező. A karakteres berendezés neve. (Blokkos berendezés esetén a handler által kezelt egységek számát tartalmazza.)

Tehát ha a BP:SI + 4. címen lévő szó 15. bit-je 0, azaz blokkos berendezésről van szó, akkor a 4. szónak csak a 13. és 14. bit-je lehet nullától eltérő. A 13. bit abban az esetben egy, ha a lemez nem IBM DOS formátumú.

Karakteres berendezés esetén a BP:SI + A. címtől kezdődő 8 byte-ról tudjuk megállapítani a berendezés nevét.

A DHC részletes ismertetése a felhasználó által írt perifériakezelőkkel foglalkozó fejezetben található.

16.6 A 25-ös megszakítás: közvetlen lemezolvasás

A vezérlés közvetlenül a DOS BIOS-ra kerül. Visszatéréskor az eredeti flag-ek a verembcn vannak, mivel azokat az INT utasítás lementette. Ezért visszatérés után a program jellemzőket vissza kell állítani. Az interrupt hívás bemenő paraméterei:

AL	A meghajtó azonosítója. (0 = A, 1 = B, 2 = C stb.)
CX	Az olvasandó szektorok száma.
DX	Az első szektor logikai száma.
DS:BX	Átviteli terület cím. (DTA)

A logikai szektorszám a szektorok következőképpen értelmezett egymásutáni elhelyezkedéséből adódik: a 0-ás sávon, a 0-s fejjel elérhető, fizikailag 1-es szektor a logikai 0. szektor, 0. fejjel elérhető fizikailag 2. a logikai 1. stb., majd a 0. fej utolsó (8. vagy 9.) szektorát kétoldalas lemeznél az egyes fej (lemezoldal) 0. sávjának első szektora követi, és így folytatódik a számozás az utolsó sáv utolsó fejének utolsó szektoráig. Ezért, bár a szektorok növekvő sorrendben vannak számozva, fizikailag nem biztos, hogy egymás mellett állnak. A 25-ös megszakítás meghívása esetén a szegmens regiszterek kivételével valamennyi regiszter tartalma törlődik. Ha az átvitel sikeres volt, akkor a CF = 0 (Carry Flag), ellenkező esetben CF = 1.

Hiba esetén az AX a hibára mutat. Az AL a DOS-hibakódját tartalmazza, amely azonos azzal, amelyet a DI alacsonyabb értékű byte-jában akkor kapunk, ha INT 24-et hív a DOS.

Az AH regiszter pedig hiba esetén a következő értékeket tartalmazhatja (mint mindig, hexadecimálisan értve):

80	A meghajtó üres, nincs benne floppy.
40	A pozicionálás hibás.
20	Vezérlőegység hiba.
10	Hibás CRC a lemez olvasásnál.
08	DTA-túllépés a műveletnél.
04	A kért szektor nem található.
03	Írásvédett lemezre próbáltunk írni.
02	Egyéb hiba.

16.7 A 26-os megszakítás: fizikai szintű lemezírás

Ami a 25-ös megszakításra vonatkozott, az itt is érvényes, csak a műveletnél olvasás helyett mindenütt írás értendő.

16.8 A 27-es megszakítás: befejezés, rezidens programok részére

Ezt azok a programok használják, amelyek különböző okok miatt lefutásuk után is a memóriában maradnak. Ehelyett jobb a 31-es DOS funkciót használni, mert azzal 64 Kbyte-nál nagyobb program is a memóriában maradhat, és RETURN (ERROR CODE vagy COMPLETION CODE) információk is átadhatók a DOS számára.

A 27-es interrupt meghívása előtt a programnak a DX-be kell tölteni az ő utolsó címe utáni cím értékét, ez offset érték, DS-hez viszonyítva, és DS = ES kell legyen. A DOS a 27-es megszakítással befejezett programot úgy kezeli, mint a DOS egy kiterjesztését, így ez a program akkor sem íródik felül, ha más programokat töltünk be és hajtunk végre. Így például felhasználó által írt megszakítás-kezelő programok esetén, amelyeknek állandóan a memóriában kell lenni.

28-2E megszakítások: a DOS számára fenntartottak

16.9 2F-es megszakítás: multiplex megszakítás

A 2F megszakítás a DOS olyan külső parancsaival való kapcsolattartásra szolgál, amelyek az első meghívásuk után a memóriában maradnak. (PRINT, ASSIGN, SHARE, APPEND). Ezen túlmenően a felhasználó szabadon definiálhatja a megszakítás használatát.

Minden egyes funkcióhoz tartozik egy ún. multiplexer szám. Az egyes funkciókat láncba kell fűzni, azaz amikor saját multiplexer programot írunk, az az inicializáló részben gondoskodik a 2F megszakítási vektor átírásáról. A rezidens rész, ha megkapja a vezérlést, ellenőrzi, hogy neki szóló multiplexer számot kapott-e. Ha nem, akkor átadja a vezérlést a lánc következő tagjának. Ha igen, akkor elvégzi a megfelelő feladatot.

A DOS-ba beírt funkciók úgy vannak elkészítve, hogy az AH-ba mindig a multiplexer számot míg az AL-be az elvégzendő feladat kódját kell beírni. A jelenleg érvényes párosítások az alábbiak:

PRINT rezidens része (AH = 1)

A multiplexer program a következő feladatokat tudja elvégezni:

AL = 0 *Installálási állapot lekérdezése.*

Visszatéréskor az AL arra vonatkozó információt tartalmaz, hogy a PRINT rezidens részét installáltuk-e már:

- AL = 0 még nem, de lehetséges
- AL = 1 még nem, és nem is lehet
- AL = FF installált.

AL = 1 *File leírása a nyomtatási sorba*

DS:DX a megszakítás meghívása előtt egy byte-ból és egy mutatóból álló memóriarészre mutat. A byte a verziót írja le (DOS 3.10-3.30 esetén kötelezően 0). A duplaszó eltolás, szegmens sorrendben a file-t definiáló ASCIIZ string elejére mutat. A file-név nem tartalmazhat * és ? karaktereket.

AL = 2 *File(ok) törlése a nyomtatási sorból*

DS:DX a sorból törlendő file-név elejére mutat. Az ASCIIZ string tartalmazhat * és ? karaktereket is.

AL = 3 *A nyomtatási sor törlése*

A funkció valamennyi file-t törli a nyomtatási sorból.

AL = 4 *Állapot lekérdezés kezdete*

Vissszatéréskor DX a hibák számát tartalmazza. Ez a PRINT sikertelen kísérleteinek számát mutatja (miközben az utolsó karaktert nyomtatja). Ha DX = 0, akkor nem volt hiba. A funkció felfüggeszti a PRINT működését. A DS:SI a nyomtatási sorra mutat. A sor pontosan 64 byte hosszú bejegyzéseket tartalmaz, amelyek a nyomtatandó file-neveket tartalmazzák. Az első bejegyzés az éppen nyomtatott file neve. A sor végét egy olyan bejegyzés jelzi, melynek első karaktere 0.

AL = 5 *Állapot lekérdezés után továbbindítás*

A hívás hatására a sor nyomtatása - amit az AL = 4 hívás felfüggesztett - folytatódik. Visszatéréskor AX a hiba kódját tartalmazza. Az egyes kódok jelentése a következő:

AX	Jelentés
1	érvénytelen funkció
2	a file nem létezik
3	az út nem létezik
4	túl sok nyitott file
5	az elérés tilos
8	a sor megtelt
9	az eszköz foglalt
12	a név túl hosszú

AL = F8-FF: *a DOS számára fenntartott*

ASSIGN rezidens része (AH = 2)
SHARE rezidens része (AH = 10)
APPEND rezidens része (AH = 67)

Az AL = 0 esetben a rezidens rész installálásáról kapunk információt, úgy, mint a PRINT esetén. Egyéb funkció nincs.

17. DOS funkciók

A DOS a perifériák (lemezegységek és az összefoglaló néven karakteres berendezéseknek nevezett klaviatúra, display és nyomtató) I/O műveleteit, file-ok kezelését, a memória kezelését, dátum- és időbeállítását és egyéb műveleteket ellátó programokhoz, az úgynevezett DOS funkciókhoz biztosítja a felhasználó hozzáférését. Az alábbiakban megadjuk e funkciók hívási számát (hexadecimálisan) és tudnivalóit. A DOS funkciók csoportjai:

I/O műveletek karakteres berendezésekkel

01 = tasztatúra input
02 = kiírás képernyőre
03 = soros vonali input
04 = soros vonali output
05 = karakter kiírása nyomtatóra
06 = közvetlen konzol I/O
07 = közvetlen konzol input, képernyőre írás nélkül
08 = konzol input, képernyőre írás nélkül
09 = karaktersorozat kiírása
0A = pufferelt tasztatúra input
0B = standard input egység állapotának ellenőrzése
0C = a standard input puffer törlése, és egy standard input kezelő rutin hívása

Processzor-kezelő funkciók

00 = program befejezése (helyette inkább a 4C-t használjuk!)
26 = új program szegmens előállítása (helyette inkább 4B-t használjuk!)
31 = programfutás befejezése úgy, hogy a program a memóriában marad
4B00 = program betöltése és végrehajtása (EXEC)
4B03 = program betöltése végrehajtás nélkül
4C = program befejezése
4D = a hívott program befejezési kódjának lekérdezése (WAIT)
62 = program szegmens lekérdezése

Memória kezelés

48 = memória foglalás
49 = lefoglalt memória felszabadítása
4A = lefoglalt memóriablokk méretének megváltoztatása

Tartalomjegyzékkel kapcsolatos műveletek

39 = altartalomjegyzék létrehozása (MKDIR)
3A = altartalomjegyzék törlése (RMDIR)
3B = aktuális altartalomjegyzék beállítása (CHDIR)
41 = file törlése (UNLINK)

43 = file attributumának lekérdezése vagy beállítása
47 = adott meghajtó aktuális altartalomjegyzékének lekérdezése
4E = elérési úttal megadott file keresése az első találatig
4F = további, a megadottnak megfelelő file keresése (4E után)
56 = file átnevezése
57 = file dátumának és időpontjának lekérdezése, illetve módosítása

FCB-s file-kezelő funkciók (használatuk ellenjavalt!)

0F = file megnyitása
10 = file lezárása
11 = adott FCB-beli file-név keresése a tartalomjegyzékben
12 = a 11-es funkcióval megtalált file-t követő további egyező file keresése
13 = file törlése
14 = file soros olvasása
15 = file soros írása
16 = file létrehozása
17 = file átnevezése
21 = random olvasás
22 = random írás
23 = file-méret lekérdezése
24 = relatív rekord mező beállítása
27 = random blokk olvasása
28 = random blokk írása

File-számos file-kezelés

3C = file létrehozása, vagy meglevő file hosszának 0-ra állítása
3D = file megnyitása
3E = file lezárása
3F = olvasás file-ból vagy egyéb perifériáról
40 = irás file-ba vagy egyéb perifériára
42 = írási/olvasási mutató állítása
45 = file file-számának duplikálása
46 = file-szám duplikálása úgy, hogy a másodpéldányt a felhasználó adja meg
5A = file létrehozása egyedi névvel
5B = új file létrehozása
67 = file-számos periféria-kezelők maximális számának megadása
68 = puffer kiürítése

Egyéb rendszerfunkciók

0D = lemez alaphelyzetbe állítása
0E = lemezegység kiválasztása
19 = az akuális meghajtó kódjának bekérése
1A = a DTA címének beállítása
1B = az aktuális meghajtó információinak bekérése
1C = adott meghajtó információinak bekérése

25 = a megszakítási (interrupt) vektor beállítása
29 = file-azonosító keresése adott parancssorban
2A = dátum lekérdezése
2B = dátum beállítása
2C = rendszeridő lekérdezése
2D = rendszeridő beállítása
2E = VERIFY (lemezírás utáni ellenőrzés) kapcsoló beállítása
2F = lemez átviteli terület (DTA) címének lekérdezése
30 = DOS verziószám lekérdezése
33 = Ctrl-Break lekérdezése és beállítása
35 = megszakítás (interrupt) vektor lekérdezése
36 = szabad lemezterület lekérdezése
38 = ország-specifikus információk lekérdezése
54 = VERIFY (lemezírás utáni ellenőrzés) kapcsoló lekérdezése
65 = kibővített ország-specifikus információk lekérdezése
66 = globális kódlap beállítása/lekérdezése

File megosztás és hálózatkezelés

3D = file megnyitása adott elérési móddal
4409 = átirányítás ellenőrzése blokkos berendezésre
440A = átirányítás ellenőrzése karakteres berendezésre
440B = elérési kísérletek számának beállítása
5C00 = file zárolása
5C01 = file zárolásának feloldása
5E00 = lokális név lekérdezése
5E02 = nyomtató inicializálása
5F02 = hozzárendelés lekérdezése
5F03 = hozzárendelés beállítása
5F04 = hozzárendelés törlése

A 2F-57 funkciók a DOS 2.00-ban jelennek meg. Azokban az esetekben, amikor hasonló funkciók kisebb és nagyobb hívószámmal is szerepelnek, akkor az újak (nagyobb számúak) használatát javasoljuk. Ezek definíciója egyszerűbb és így teljesítményük jobb, mint a hagyományos párjainak. Ha a programban az új funkcióhívásokat használjuk, akkor azokat a DOS 1.00 vagy 1.10 operációs rendszerben nem lehet végrehajtani.

A DOS 3.00-ás verziótól lehet az 55-5F funkciókat használni, amelyek a hálózatkezeléssel foglalkoznak.

Ha a DOS a vezérlést átveszi, akkor egy belső veremre kapcsol át. A felhasználói regisztereket ebbe menti, azért, hogy az igényelt függvényhívás az információkat vissza tudja adni.

A felhasználói területen lévő veremnek elég nagynak kell lennie ahhoz, hogy a megszakítási rendszer az összes regisztert le tudja menteni.

Azt ajánljuk, a felhasználói veremben azon kívül, amit a felhasználó akar benne tárolni, legyen legalább 80 (hexadecimális) byte-nyi hely a DOS számára.

17.1 A DOS funkciók hibakezelése

Ha a művelet sikeresen befejeződött, akkor az előbb említett új funkcióhívások többsége a flag-et törölt átviteli bit-tel (Any Flag) adja vissza (No Carry), hiba esetén pedig beállítja az átvitelt (Carry Yes). Ez utóbbi esetben az AX a következő bináris hibakódokat tartalmazhatja:

Kód	Leírás
1	Érvénytelen függvényhívás
2	A file nem található
3	Az elérési út nem található
4	Túl sok megnyitott file
5	Az elérés nincs engedélyezve (hálózati alkalmazásoknál)
6	Érvénytelen kezelés
7	A memóriakezelő blokkok lánca megsérült
8	Nincs elég memória
9	Érvénytelen memória-blokk cím
10	Érvénytelen környezet
11	Érvénytelen forma
12	Érvénytelen elérési kód
13	Érvénytelen adatok
15	Érvénytelen meghajtó megadás
16	Kísérlet az aktuális könyvtár törlésére
17	Nem ugyanaz az egység
18	Nincs több file
19	Írásvédett lemez
20	Rossz lemezegység
21	A lemezegység nincs készenléti állapotban
22	Érvénytelen lemezműveleti parancs
23	CRC hiba
24	Nem megfelelő hossz
25	Keresési (seek) hiba
26	Nem DOS lemez
27	Szektort nem találja
28	Kifogyott a papir
29	Iráshiba
30	Olvasási hiba
31	Általános hiba
32	Megosztás nem megengedett
33	Zárolás nem megengedett
34	Rossz lemez
35	FCB nem elérhető
36	puffer túlcsordulás a megosztásnál
37-49	Fenntartott
50	Nincs hálózat
51	A távoli számítógép nem 'hallgató'

Kód	Leírás
52	Dupla név a hálózatban
53	A hálózati nevet nem találja
54	A hálózat foglalt
55	A hálózaton implementált eszköz nem létezik már
56	A hálózati BIOS határát túlléptük
57	Hálózati adapter hiba
58	Hibás válasz a hálózatból
59	Felismerhetetlen hálózati hiba
60	Nem kompatibilis hálózati adapter
61	A nyomtatási sor megtelt
62	A nyomtatási sor nincs tele
63	Nincs elég hely a nyomtatási file-hoz
64	A hálózati nevet törölték
65	Az elérés nincs megengedve
66	A hálózati eszköz típus nem megfelelő
67	A hálózat neve nem létezik
68	A hálózati név határát túlléptük
69	A hálózati BIOS szelet határát túlléptük
70	Ideiglenesen szünetel
71	A hálózati kérés nem szolgálható ki
72	A lemez vagy nyomtató átirányítása szünetel
73-79	Fenntartott
80	A file létezik
82	Nem hajtható végre
83	24-es típusú megszakítási hiba
84	A struktúrán kívül van
85	Már hozzárendelték
86	Érvénytelen jelszó
87	Hibás paraméter
88	Hálózati adathiba

Az 1-18 hibák a DOS 2.10-es változatától szerepelnek, s azonosak az INT 21 megszakítás által kezelt hibákkal. Hasonlóan a 19-31 sorszámú hibák azonosak az INT 24 megszakítás által kezelt 0-12 sorszámú hibákkal. A 32-83 hibákat a DOS 3.00-ás verzióitól kezdve jelenti a rendszer, s elsősorban a file-megosztással és a hálózatkezeléssel kapcsolatosak. A 84-88 hibakódok a 3.10-es verzióban jelennek meg először.

17.2 A DOS funkcióhívások általános szabályai

A kívánt funkció számát mindig az AH regiszterbe kell tenni. Ez a szám az a hexadecimális szám, amellyel az egyes funkciókat itt le fogjuk írni. Egyes funkciók más regiszter beállítását is megkívánják, ezeket, és az esetleges kimeneti regisztertartalmakat az illető funkciók leírásánál adjuk majd meg.

Egyes funkcióhívások inputja ASCII jelsorozat is lehet. Ebben az esetben az ASCII jelsorozat meghajtó-, elérési út- és néhány esetben file-névből áll. A jelsorozat egy bináris

0-t tartalmazó byte-tal fejeződik be. Figyelemmel kell lenni arra, hogy azok a funkcióhívások, amelyek útmegadást elfogadnak, elválasztóként a \-t (backslash) használják.

Azok az új funkcióhívások, amelyek a file-kezeléssel foglalkoznak, használnak egy numerikus azonosítót, amelyet "file-szám"-nak (handle) nevezünk. Ha ezekkel állítunk elő, vagy nyitunk meg file-t, vagy egységet, akkor az AX tartalma egy 16 bites bináris érték lesz, amelyet file-szám-ként kapunk vissza. Ez a file-szám azonosítja a megnyitott file-t. A DOS-ban a következő file-számok fenntartottak:

0000	Standard input egység (az input átirányítható).
0001	Standard output egység (az output átirányítható).
0002	Standard hibakiíró-egység (az output nem irányítható át).
0003	Standard vonali egység.
0004	Standard nyomtatóegység.

Ezeket megnyitás nélkül lehet a felhasználói programokban használni.

A legtöbb funkció hívásakor a paramétereket regiszterekben kell átadni. Az egyes funkcióknál ismertetjük majd, hogy mely regiszterek játszanak szerepet. Az esetleges paraméterbeállítás után a funkciók hívása a következő eljárások valamelyikével történik:

1. A funkciót azonosító értéket az AH-ban kell beállítani. A program szegmens prefixben (PSP) hexadecimális 50-re adjunk ki egy CALL FAR szubrutinhívást. (Itt figyelembe kell venni azt, hogy azok a programok, amelyek ezeket a funkcióhívásokat használják, a DOS 1.00 és az 1.10 operációs rendszer alatt nem működnek jól.)

2. A funkcióhívás azonosító értékét az AH-ban beállítjuk, majd kiadjuk a 21-es megszakítást (INT 21).

3. Léteznek programok, amelyeket más hívási konvenciókkal írtak meg. Ezek a funkció számát a CL-be teszik, a többi regisztert a funkcióhívás specifikációjának megfelelően állítják be. Ezután a PSPA + 5-re egy közeli CALL-t adnak ki. Ezen a címen egy CALL FAR van, amely a DOS funkciókat adminisztráló programot hív meg. Ennél a módszernél az AX-regiszter tartalma mindig törlődik. (CP/M konvenció. Használata ellenjavalt!)

Megjegyzés:

1. A funkcióhívásokra általánosan igaz:
   - A funkcióhívás kódját az AH regiszterben kell megadni;
   - A memóriaterületek címét a DS:DX regiszterekben kell megadni;
   - A funkcióhívás eredménye az AX regiszterbe kerül.
2. Az assembly, 'C' és a PASCAL nyelvből közvetlenül is meg lehet a funkciókat hívni. A funkcióhívásokat viszont közvetve minden magasszintű nyelvből elérhetjük, ha az adott programozási nyelvből meghívjuk a megfelelő assembler programot.
3. A FCB-vel foglalkozó funkcióhívások speciális karaktereket (0D-29 hexadecimális) nem engednek meg.

4. A funkcióhívások az AX regiszter tartalmát megváltoztathatják, így arra ne számítsunk, hogy ott változatlan érték marad.

5. A funkcióhívásokra vonatkozó példákat Turbo Pascal-ban adjuk meg. Ennek oka, hogy ez sokkal könnyebben olvasható, mintha pl. MACRO ASSEMBLER-ben adnánk meg. A példák megadásakor feltételezzük, hogy a Turbo Pascal program az alábbi deklarációkat tartalmazza:

```
type
  RegisterSet = record case integer of
                1: (AX,BX,CX,DX,BP,DI,SI,DS,ES,Flags: integer);
                2: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
                end {record};
   buffer       =array[1..32] of byte;
   szoveg       =string[255];
   mutato       =^buffer;
   integer2     =record
                     LowInt,HighInt: integer
                  end;
   fcbset         =record
                     Drive: byte;
                     Name: string[11];
                     Extention: string[3];
                     CurrBlock: integer;
                     RecodSize: integer;
                     FileSize: integer2;
                     Date: string[2];
                     Time: string[2];
                     Reserved: string[8]
                     CurrRecord: byte;
                     RelRecord: integer2
                  end;
 var reg: RegisterSet;
```

17.3 A DOS funkciók ismertetése

17.3.1 0-s funkció - program befejezése

A funkció a programot úgy fejezi be, hogy az INT 22, 23 és 24 címeket visszaállítja azokra az értékekre, amelyeket a programozó a funkció hívása előtt a PSP-be kimentett.

A funkció valamennyi adatpuffert lezárja (kiüríti). Azok a file-ok, amelyeket módosítottunk, a régi jellemzőkkel kerülnek a tartalomjegyzékbe (dátum, hossz stb.). Ezért ezeket előbb le kell zárni (lásd a 10-es funkciót!). A vezérlés a befejezési címre kerül. Ez a funkcióhívás ugyanazt hajtja végre, mint az INT 20. A funkció hívása előtt gondoskodni kell arról, hogy a CS regiszterbe kerüljön a PSP szegmenscíme. Ez a funkció egyik regiszterben sem ad vissza információt. (A DOS-hoz kerül a vezérlés.)

17.3.2 1-es funkció - tasztatúra input

A standard input egységről karakter beírását várja, majd ezt a jelet a standard output egységen is visszaírja (echo). Visszatéréskor a karakterkód az AL regiszterbe kerül. Ha ez a karakter éppen a < Ctrl-Break >, akkor a 23-as megszakítás is végrehajtódik.

Megjegyzés:

Az 1-es, 6-os, 7-es és 8-as rendszerfunkciók esetén a kibővített ASCII kódok két funkcióhívást igényelnek. Az első hexadecimális 0-t ad vissza, amellyel jelzi a rendszer, hogy kiterjesztett kód következik.

Példa:

1.

```
function KbdEcho: char;
begin
   reg.AH:=$01;
   MSDos(reg);
   KbdEcho:=chr(reg.AL)
end;
```

2.

```
procedure ExtKbdEcho (varC: char,var E: boolean)
begin
 C:=KbdEcho;
 if C=#0 then
 begin
   C:=KbdEcho;
   E:=true
 end
 else E:=false
end;
```

A KbdEcho függvény azonos a Turbo Pascal 4.0-ás verziójában levő ReadKdb függvénnyel.

17.3.3 2-es funkció - kiírás képernyőre

A DL-ben lévő karakter a standard outputra kerül. Ha a karakter a < balranyíl >, akkor a kurzor egy hellyel balra lép és erre a helyre egy szóköz jel kerül.

A funkció a 09(TAB), 0A(LF) és 0D(CR) kódokat megfelelően értelmezi.

Példa:

```
procedure DisplayChr(ch:char);
begin
   reg.DL:=ord(ch);
   reg.AH:=$02;
   MSDos(reg)
end;
```

17.3.4 3-as funkció - soros vonali input

A standard soros vonalról vár egy karaktert. A jelet az AL regiszterben kapjuk meg.

Megjegyzés:

1. A soros vonalak (AUX, COM1, ..., COM4) pufferolás nélkül hajtják végre az I/O műveleteket.
2. Indításkor a DOS az első soros vonalat a következő kezdeti állapotra állítja be: 2400 baud, paritás nélkül, 1 stop-bit, 8 adat-bit.
3. A soros vonalak rendszerhívásai (3, 4) állapot- és hibakódot nem adnak. Az ellenőrzés érdekében ezért javasoljuk a ROM BIOS rutin (INT 14) használatát.

Példa:

```
function ReadAux: byte;
begin
   reg.AH:=$03;
   MSDos(reg);
   ReadAux:=reg.AL
end;
```

17.3.5 4-es funkció - soros vonali output

A DL-ben lévő jel a standard soros vonalra kerül. Ez a funkció egyik regiszterben sem ad vissza információt.

Példa:

```
procedure WriteAux(ch: byte);
begin
   reg.DL:=ch;
   reg.AH:=$04;
   MSDos(reg)
end;
```

17.3.6 5-ös funkció - karakter kiírása nyomtatóra

A DL-ben lévő karakter a standard nyomtatóra kerül. Ez a funkció egyik regiszterben sem ad vissza információt.

Példa:

```
procedure PrintChar (ch:char);
begin
   reg.DL:=ord(ch);
   reg.AH:=$05
   MSDos(reg)  .
end;
```

17.3.7 6-os funkció - közvetlen konzol I/O

Az input és az output között a DL regiszter tartalmának beállításával választhatunk.

- FF - input
- bármi más - output

Konzol input funkció végrehajtása után, ha van az input pufferban várakozó karakter, a Zero Flag törlődik (clear) és AL-ben megkapjuk a beolvasott karaktert. Ha nincs várakozó karakter, a Zero Flag értéke 1 lesz (set).

Konzol output funkció hívása előtt DL-be be kell tenni a standard outputra kiírandó karaktert.

A funkció nem figyeli sem a <Ctrl-Break>-et, sem a <Ctrl-PrtSc>-t.

Kibővített ASCII kódú billentyű lenyomásának értékelésére két funkcióhívásra van szükség. Az első esetben #0 karaktert ad vissza a funkció.

Példa:

```
procedure ConsolIO(command:byte;
                   var ch: byte;
                   error:  boolean);
begin
   reg.DL:=command;
   reg.AH:=$06;
   MSDos(reg);
   if (command=$ff) and (reg.Flags and 1)=1 then error:=true
   else begin
      ch:=reg.AL;
      error:=false
   end
end;
```

17.3.8 7-es funkció - közvetlen konzol input, képernyőre írás nélkül

A standard inputról vár egy karaktert, amelyet az AL-ben továbbít. A billentyű ellenőrzés ugyanaz, mint a 6-os funkciónál.

Kibővített ASCII kódok esetén két funkcióhívásra van szükség. Az első #0 kódú karakterrel tér vissza.

Példa:

```
function ReadConsole: byte;
begin
   reg.AH:=$07;
   MSDos(reg);
   ReadConsole:=reg.AL
end;
```

17.3.9 8-as funkció - konzol input képernyőre írás nélkül

Karaktert olvas be AL-be a standard inputról, mint a 7-es funkció, de nem írja vissza. < Ctrl-Break > megnyomása esetén a 23-as megszakítást is végrehajtja.

Példa:

```
function InpKbd: byte;
begin
   reg.AH:=$08;
   MSDos(reg);
   InpKbd:=reg.AL
end;
```

17.3.10 9-es funkció - karaktersorozat nyomtatása

Hívásakor a DS:DX egy olyan karaktersorozatra mutat a memóriában, amely dollárjellel ($: hexadecimális 24) fejeződik be. A sorozat minden jele a standard outputra kerül (a dollárjel már nem).

Ez a funkció egyik regiszterben sem ad vissza információt.

Példa:

```
procedure DisplayStr(irat: szoveg);
begin
   irat:=irat+'$';
   reg.DS:=Seg(irat);
   reg.DX:=Ofs(irat)+1;
   reg.AH:=$09;
   MSDos(reg)
end;
```

17.3.11 A funkció - pufferelt tasztatúra input

A DS:DX az input pufferra mutat. Az input puffer első byte-ja nem lehet nulla, hanem a pufferben maximálisan tárolható karakterek számát adja meg. A funkció a karaktereket a standard input egységről olvassa és a puffer 3. byte-jától kezdve helyezi el. A standard inputról az átvitel mindaddig folytatódik, amíg az <Enter> billentyűt le nem nyomjuk. Ha már csak egy karakter fér el a pufferbe és még további karaktereket akarunk olvasni, akkor karakterenként egy-egy hangjelzés figyelmeztet arra, hogy az <Enter>-t nyomjuk le, és ezeket a karaktereket nem olvassa be. A puffer második byte-ja a ténylegesen átvitt karakterek számát adja meg. Az "üzenet vége"-jel (carrige return 0D) nem tartozik a jelsorozathoz, ez mindig az utolsó jel. A puffer felépítését a későbbiekben ismertetjük.

Példa:

```
procedure ReadLine(MaxLength: byte;
                   var irat: szoveg);
begin
   irat[0]:=chr(MaxLength);
   reg.DS:=Seg(irat);
   reg.DX:=Ofs(irat);
   reg.AH:=$0a;
   MSDos(reg);
   irat:=copy(irat,2,ord(irat[1]))
end;
```

17.3.12 B funkció - standard input egység állapotának ellenőrzése

Ha a standard input egységről egy karakter átvihető, akkor (AL)=FF, különben (AL)=00. <Ctrl-Break> esetén 23-as megszakítás történik.

Példa:

```
function IsEmptyKbd: boolean;
begin
   reg.AH:=$0b;
   MSDos(reg);
   if reg.AL=$ff then IsEmptyKbd:=false
   else IsEmptyKbd:=true
end;
```

17.3.13 C funkció - a standard input puffer törlése, és egy standard input kezelő funkció hívása

A standard input pufferbe bevitt valamennyi karakter törlődik, majd az AL-ben megadott sorszámú funkció kerül végrehajtásra. (Csak 1, 6, 7, 8 és A megengedett.) A rendszer addig vár, amíg legalább egy karaktert be nem olvas.

A funkció az alfunkciónak megfelelően a beolvasott karakter(eke)t vagy állapotot adja vissza.

Példa:

```
procedure FlushAndRead(command: byte;
                       var mode: byte);
begin
   reg.AL:=mode;
   reg.AH:=command;
   MSDos(reg);
   mode:=reg.AL
end;
```

17.3.14 D funkció - lemez alaphelyzetbe állítása

A funkció inicializálja a lemezegységet, a speciális file puffereket kiüríti. A módosított file-ok, melyeknek jellemzői (dátum, méret stb.) megváltoztak, hibásan maradnak a tartalomjegyzékben. Lemezcsere esetén, ha az összes olyan file-t, amelybe írtunk, lezártuk, nem kell hívnunk ezt a funkciót.

Ez a funkció egyik regiszterben sem ad vissza információt.

Példa:

```
procedure ResetDisk;
begin
   reg.AH:=$0d;
   MSDos(reg)
end;
```

17.3.15 E funkció - lemezegység kiválasztása

A DL-ben megadott meghajtót (0 = A, 1 = B stb.) teszi aktuális meghajtóvá. A meghajtók számát az AL-be írja (floppy egységek száma + fix lemezek száma). Ha a számítógép csak egy meghajtóval rendelkezik, a rendszer akkor is két meghajtót feltételez, az A: és a B: logikai meghajtókat. (A floppy egységek tényleges számának meghatározására használhatjuk az INT 11-et is, amellyel a BIOS állapítja meg az egységek jelenlétét.)

Példa:

```
procedure SetDisk(drive: byte;
                  var MaxDrive: byte);
begin
   reg.DL:=drive;
   reg.AH:=$0e;
   MSDos(reg);
   MaxDrive:=reg.AL
end;
```

17.3.16 F funkció - file megnyitása

A DS:DX egy olyan FCB-re mutat, amelyet meg kell nyitni. Az aktuális tartalomjegyzékben megkeresi a megadott file-t. Ha nem találta meg, akkor az AL-ben FF-et ad vissza. Ha megtalálta, akkor az AL-be 00-t ír és az FCB-t a következő módon tölti fel:

Ha a meghajtókód = 0 (aktuális meghajtó), akkor ezt az aktuális meghajtó kódjára állítja (1 = A, 2 = B stb.). Így az aktuális meghajtó anélkül változtatható, hogy az ezzel a file-lal végzendő műveleteket befolyásolná. Az FCB aktuális blokkot megadó mezőjét nullára állítja. A feldolgozandó rekord hosszát hexadecimális 80-ra állítja. Ha más hosszt akarunk, ezt az értéket nekünk kell megváltoztatnunk. Ha random file-kezelésre készülünk, az FCB-nek a random file-kezeléssel foglalkozó byte-jait is állítani kell (lásd az FCB részletes leírását). Az FCB esetleges változtatását a megnyitás után, de a többi művelet előtt kell végrehajtani.

Példa:

```
procedure OpenFile(var fcb: fcbSet;
                   var error: boolean);
begin
   reg.DS:=Seg(fcb)
   reg.DX:=Ofs(fcb);
   reg.AH:=$0f;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.17 10-es funkció - file lezárása

A file módosítása után végre kell hajtani, hogy helyes információk kerüljenek a tartalomjegyzékbe. A DS:DX egy megnyitott FCB-re mutat. A funkció a lemez aktuális tartalomjegyzékét átvizsgálja. Ha megtalálja a file-t, akkor helyzetét összehasonlítja a file megnyitásakor a tartalomjegyzékben talált helyzetinformációkkal. Ha ezek nem egyeznek, akkor feltételezi, hogy a floppy-t kicserélték. Ebben az esetben, vagy ha a file-t az aktuális tartalomjegyzékben nem találja, az AL-be FF-et ír, egyébként 00-t és módosítja is a lemez tartalomjegyzékét az FCB-ben talált információnak megfelelően.

Példa:

```
procedure CloseFile(fcb: fcbSet;
                    var error: boolean);begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$10;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.18 11-es funkció - adott FCB-beli file-név keresése tartalomjegyzékben

A DS:DX-ben megadott, nem nyitott FCB-hoz a DOS megkeresi a file-név első előfordulását. A név tartalmazhat globális név-karaktereket ($ és *) is. Ezt a file-nevet aztán megpróbálja az aktuális meghajtó tartalomjegyzékében is megtalálni. Ha nem talál ilyent, AL-be FF-et tesz. Ha talál, AL-be 00-t tesz, és a megtalált file további adatait a lemez átviteli területre (DTA) olvassa.

Aszerint, hogy a DS:DX-ben megadott FCB kibővített, vagy normál, a DTA-ba sikeres keresés után a következő információ kerül:

Ha az FCB kibővített, akkor a 11-es funkció a DTA első byte-ját FF-re, az azt követő 5 byte-ot 0-ra, az ezt követőt a megadott FCB kereséséből származó attributum byte-ra állítja, majd leírja (a szokott módon: 1 = A, 2 = B stb.) a meghajtót, ezután teszi a 32 byte-os tartalomjegyzék bejegyzést (directory entry). Így a DTA-ba egy érvényes kiterjesztett FCB kerül, amely egy nem megnyitott file-hoz tartozik, s ugyanaz az attributum byte-ja, mint a funkció számára megadott, keresett FCB-é.

Ha a keresésnél használt FCB normál, akkor az első byte a meghajtó száma (a szokott módon: 1 = A, 2 = B stb.), az ezt követő 32 byte a megtalált file-névhez tartozó tartalomjegyzék bejegyzés. Így a DTA-ba egy érvényes, normál, nem megnyitott FCB kerül.

Ha kibővített FCB-t adtunk meg, a keresést az attributum-byte értéke határozza meg:

1. Ha az FCB attributum byte-ja nulla, akkor a keresés a kötetazonosítót (volume label), az altartalomjegyzékeket, a rejtett és a rendszer-file-ok bejegyzéseit nem találja meg.

2. Ha az attributum-byte rejtett-, vagy rendszer-file-t jelez, akkor a funkció az egyező normál file-okon kívül ezeket is megtalálja. Ha minden egyező tartalomjegyzék bejegyzést meg akarunk kapni, kivéve a kötetazonosítót, az attributum-byte mindhárom bitjét 1-re kell állítani (lásd a lemez tartalomjegyzék leírását).

3. Ha az attributum-byte a kötetazonosítóra van állítva, akkor részleges keresés történik, és csak az egyező adathordozó-azonosítót találja meg.

Példa:

```
procedure FirstEntry(fcb: fcbSet;
                     var error: boolean);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$11;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.19 12-es funkció - a 11-es funkcióval megtalált file-t követő további egyező file keresése

Ha a 11-es funkció talált egy, a megadottal egyező bejegyzést, akkor a 12-es segítségével megkereshetjük a következő egyező bejegyzést. Megengedett a file-azonosítóban a globális név-jelek (? és *) használata. Az input és output regiszterek értéke azonos a 11-es funkcióéval. Az FCB fenntartott területén azok az információk állnak, amelyek a keresés folytatásához szükségesek. (Több file keresésénél ezt az FCB-t ne használjuk file-műveletekhez!)

Példa:

```
procedure NextEntry(fcb: fcbSet;
                    var error: boolean);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$12;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.20 13-as funkció - file törlése

A DS:DX egy nem megnyitott FCB-re mutat. A DOS az aktuális tartalomjegyzékből valamennyi, a megadottal egyező bejegyzést törli. Ha nincs egyező, akkor az AL regiszter FF-et, különben 00-t tartalmaz. A megadott file nem lehet "READ ONLY" attribútumú. A file-név a ? globális név-jelet tartalmazhatja.

Példa:

```
procedure DeleteFile(fcb: fcbSet;
                     var error: boolean);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$13;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.21 14-es funkció - file soros olvasása

A DS:DX egy megnyitott FCB-re mutat. Az ebben meghatározott aktuális blokkszámmal és aktuális rekordszámmal címzett rekordot (lásd az FCB leírását) a funkció a DTA-ba olvassa, és aztán az aktuális rekord számát eggyel növeli. Ez a soros, másnéven szekvenciális olvasás.

Az AL értéke 0, ha az olvasás sikerült; 01, ha az olvasás EOF-ot talált és adatot egyáltalán nem tudott beolvasni; 02, ha a DTA hossz nem volt elég az olvasáshoz (szegmenshatár átlépés lenne); 03, ha EOF-ot talált, és csonka rekordot olvasott.

Példa:

```
procedure SeqRead(fcb: fcbSet;
                  var mode: byte);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$14;
   MSDos(reg);
   mode:=reg.AL
end;
```

17.3.22 15-ös funkció - file soros írása

A soros, másnéven szekvenciális írás esetén is egy megnyitott FCB-re mutat a DS:DX. A rekord file-ba kerülési helyét az FCB ugyanúgy meghatározza, mint a 14-es funkciónál.

A rekord a DTA-ból kerül a file-ba. Ha a rekord rövidebb, mint egy szektor, akkor a funkció pufferolja a rekordokat, míg egy szektornyi összegyűlik, és blokkonként írja ki.

Ezután az FCB-ben a rekord sorszámát eggyel megnöveli.

Az AL értéke 0, ha az írás sikerült; 01, ha a lemez már tele volt - és ekkor írás egyáltalán nem történt; 02, ha a DTA túl kicsi, a teljes kiírandó blokk túlnyúlna a szegmenshatáron.

Példa:

```
procedure SeqWrite(fcb: fcbSet;
                   var mode: byte);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$15;
   MSDos(reg);
   mode:=reg.AL
end;
```

17.3.23 16-os funkció - file létrehozása

A DS:DX egy nem megnyitott FCB-re mutat. A lemez aktuális tartalomjegyzékében az adott tartalomjegyzék bejegyzéssel azonos bejegyzést keres a rendszer. Ha talál ilyet, akkor ezt a bejegyzést ismételten felhasználja. Ha nem talál, akkor a következő szabad bejegyzést megkeresi a tartalomjegyzékben. Ha ilyet nem talál, akkor az AL-ben FF-et ad vissza, különben a file-t megnyitja, és az AL-be 00-t tölt.

A funkcióval rejtett file-t is létre lehet hozni kibővített FCB-vel és a megfelelő attributum byte értékkel. Ha meglevő file bejegyzést talál a DOS, a file korábbi tartalma elvész.

Példa:

```
procedure NewFileName(fcb: fcbSet;
                      var error: boolean);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$16;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.24 17-es funkció - file átnevezése

A DS:DX most egy olyan - módosított - nem megnyitott FCB-re mutat, amelyben a file-név (a régi) és a meghajtó kódja a szokott helyen áll, az új file-név pedig a régi file-név után 8 byte-tal (DS:DX + hexadecimális 11) kezdődik. A funkció, ha az aktuális tartalomjegyzékben az első file-névvel egyezőt talál, akkor annak a nevét a másodikra cseréli. (A két file azonosítója nem lehet ugyanaz.) A régi név azon pozícióiban álló karakterek, amelyeknek megfelelő helyen a másodikban "?" szerepel, változatlanul kerülnek át az új névbe.

Ha a funkció a megadott régi névvel egyező nevű file-t nem talál vagy az új név már foglalt, AL-t FF-re állítja.

Példa:

```
procedure NewFile(fcb: fcbSet;
                  var error: boolean);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$17;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.25 18-as funkció - a DOS számára fenntartott funkció

17.3.26 19-es funkció - az aktuális meghajtó kódjának bekérése

A funkció AL-ben az aktuális meghajtó kódját adja vissza. (0 = A, 1 = B stb.) Hívási paramétere nincs.

Példa:

```
function GetDisk: byte;
begin
   reg.AH:=$19;
   MSDos(reg);
   GetDisk:=reg.AL
end;
```

17.3.27 1A funkció - a DTA címének beállítása

A lemez átviteli terület (DTA) címének beállítása. A kívánt címet DS:DX-ben adjuk meg. A lemez átviteli terület maximális mérete egy szegmens (64 Kbyte). A DTA nem nyúlhat át a szegmenshatáron.

Példa:

```
procedure SetDTA(cim: integer2);
begin
   reg.DS:=cim.HighInt;
   reg.DX:=cim.LowInt;
   reg.AH:=$1a;
   MSDos(reg)
end;
```

17.3.28 1B funkció - az aktuális meghajtó információinak bekérése

A funkció segítségével a FAT (file-allokációs táblázat) információit kaphatjuk meg. A funkció hívás után a DS:DX az aktuális meghajtó FAT-jának első byte-jára mutat, amely a meghajtó típusát azonosítja. BX a lemezen lévő cluster-ek számát (lásd a FAT-tal foglalkozó szakaszt), AL a szektorok számát cluster-enként, CX a fizikai szektorméretet tartalmazza. Hívási paramétere nincs.

Példa:

```
procedure GetDeafaultDiskData(var SektorPerCluster: byte;
                              var BytesPerSector: integer;
                              var ClusterPerDrive: integer;
                              var TypeByte: byte);
begin
   reg.AH:=$1b;
   MSDos(reg);
   SectorPerCluster:=reg.AL;
   BytesPerSector:=reg.CX;
   ClusterPerDrive:=reg.DX;
   TypeByte:=mem[reg.DS,reg.BX]
end;
```

17.3.29 1C funkció - adott meghajtó információinak bekérése

A funkció hívásakor a kívánt meghajtó számát DL-ben kell megadni (a szokásos módon: 0 az aktuális, 1 az A meghajtónak stb. felel meg).

A funkció outputja ugyanaz, mint az 1B funkciónál.

Példa:

```
procedure GetDeafaultDiskData(drive: byte;
                              var SektorPerCluster: byte;
                              var BytesPerSector: integer;
                              var ClusterPerDrive: integer;
                              var TypeByte: byte);
begin
   reg.DL:=drive;
   reg.AH:=$1c;
   MSDos(reg);
   SectorPerCluster:=reg.AL;
   BytesPerSector:=reg.CX;
   ClusterPerDrive:=reg.DX;
   TypeByte:=mem[reg.DS,reg.BX]
end;
```

17.3.30 1D-20 funkciók - a DOS számára fenntartott funkciók

17.3.31 21-es funkció - random olvasás

A random (közvetlen) olvasási funkció inputként DS:DX-ben egy megnyitott FCB címét várja. Az FCB aktuális blokk és aktuális rekord mezőit megfelelően be kell állítani. Az ezekkel a mezőkkel címzett rekord a memóriában az aktuális DTA-ba kerül.

Ha az olvasás sikeresen befejeződött, a funkció AL-t 0-ra állítja. Ha file-véget (EOF) talál, akkor az AL-be 01-et vagy 03-at tesz: 01 esetén nem állt rendelkezésre több adat, 03 esetén csonka rekordot olvasott be, és a hiányzó részt 0-val töltötte fel. AL = 02 lesz, ha a DTA túl kicsi a rekord fogadására, ekkor olvasás egyáltalán nem történik.

Példa:

```
procedure RandomRead(fcb: fcbSet;
                     var mode: byte);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$21;
   MSDos(reg);
   mode:=reg.AL
end;
```

17.3.32 22-es funkció - random írás

A random (közvetlen) írási funkció inputként DS:DX-ben egy megnyitott FCB címét várja. Az FCB aktuális blokk és aktuális rekord mezőit a megfelelően ki kell tölteni. Az ezekkel a mezőkkel címzett területről írja a file-ba a rekordot a DTA-ból. (Ha az adatrekord mérete nem egyezik meg a szektor méretével, akkor pufferolt I/O műveletet hajt végre a rendszer.)

Az AL-ben visszaadott értékek jelentése:

00 - Sikeres átvitel.
01 - A lemez tele van, az írás nem történt meg.
02 - A DTA szegmenshatáron átnyúlik. Az átvitel nem történt meg.

Példa:

```
procedure RandomWrite(fcb: fcbSet;
                      var mode: byte);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$21;
   MSDos(reg);
   mode:=reg.AL
end;
```

17.3.33 23-as funkció - file-méret lekérdezése

A funkcióba lépéskor DS:DX egy nem megnyitott FCB-re mutat.

A funkció a lemez aktuális tartalomjegyzékében megkeresi a file-azonosítóval egyező első bejegyzést. Ha nem talál ilyet, akkor AL-ben FF-et ad vissza. Különben az FCB "random rekord" mezőibe beírja a file rekordjainak számát (az FCB "rekord méret" mezejében kifejezve, kerekítve), és AL-t 0-ra állítja.

Megjegyzés:

Mielőtt ezt a hívást kiadjuk, az FCB "rekord méret" mezőjét be kell állítani, különben hibás információt kapunk. (Ha a file méretére vagyunk kíváncsiak, a mezőbe 1 értéket kell írni.)

Példa:

```
procedure GetFileSize(fcb: fcbSet;
                      error: boolean);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$23;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.34 24-es funkció - random rekord mező beállítása

A DS:DX belépéskor egy megnyitott FCB-re mutat. Ez a hívás a random rekord mezőt (fcb.RelRecord) az FCB aktuális blokk és rekord mezőinek (fcb.CurrBlock, fcb.RecordSize) megfelelően állítja be.

Megjegyzés:

Random írási/olvasási (lásd 21, 22 funkciók) és random blokkírási/olvasási (lásd 27, 28 funkciók) műveletek előtt ezzel a funkcióval kell a random rekord mezőt előkészíteni.

Példa:

```
procedure SetRecord(fcb: fcbSet);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.AH:=$24;
   MSDos(reg)
end;
```

17.3.35 25-ös funkció - a megszakítási (interrupt) vektor beállítása

Inputként a funkció AL-ben a megszakítás számát, DS:DX-ben az interrupt kezelő rutin címét várja.

Regiszter-outputja a funkciónak nincs. Az adott számú interrupt-hoz az interrupt vektor táblában az adott rutincímet rendeli hozzá. (Az eredeti táblabeli érték a 35-ös funkcióval kérdezhető le.)

Példa:

```
procedure SetVector(number: byte;
                    cim: integer2);
begin
   reg.AL:=number;
   reg.DS:=cim.HighInt;
   reg.DX:=cim.LowInt;
   reg.AH:=$25;
   MSDos(reg)
end;
```

17.3.36 26-os funkció - új program szegmens előállítása

A DX belépéskor az új program szegmens szegmenscímét (paragrafusban számolva) tartalmazza.

A funkció az aktuális program szegmens 0-s címétől hexadecimális 100 hosszú területet átmásol az új szegmens 0-s címétől kezdődő területre. A 0A címtől kezdve beírja az aktuális befejezési, a Ctrl-Break kilépési-, és a kritikus hiba címeket (az interrupt vektor tábla 22-es, 23-as és 24-es megszakításokhoz tartozó értékei alapján), és az új szegmens 6-os címére beírja az új memóriaméretet.

Megjegyzés:

E funkció helyett a 4B (EXEC) funkcióhívás az ajánlott.

17.3.37 27-es funkció - random blokk olvasása

Belépéskor a DS:DX egy megnyitott FCB-re mutat. A CX az olvasandó rekordok számát tartalmazza és nem lehet nulla. A funkció a megadott számú rekordot a random rekord mezőben megadott file címről a lemez átviteli területre (DTA) olvassa.

A funkció outputja:

Ha file-vég (EOF) következik be, akkor az AL-ben 01-et vagy 03-at ad vissza aszerint, hogy nem olvasott adatot (01) vagy csonka rekordot olvasott (03). Utóbbi esetben a hiányzó részt 0-val tölti ki. Ha a lemez átviteli területre (DTA) egy olyan rekordot tölt, amelyik már a szegmenshatárt túllépné, akkor az AL-be 02 kerül, de a rendelkezésre álló területet még teljesen feltölti.

Ha valamennyi rekordot sikeresen beolvasta, akkor AL-ben 00-t ad vissza. CX mindig a ténylegesen beolvasott rekordok számát tartalmazza. Az FCB random, aktuális blokk és aktuális rekord mezőjét módosítja úgy, hogy a következő olvasandó rekordra mutassanak.

Példa:

```
procedure RandomBlockRead(fcb: fcbSet;
                          var counter: integer;
                          var mode: byte);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.CX:=counter;
   reg.AH:=$27;
   MSDos(reg);
   mode:=reg.AL;
   counter:=reg.CX
end;
```

17.3.38 28-as funkció - random blokk írása

Ugyanúgy működik, mint a 27-es funkció, csak itt írásról van szó, azaz DS:DX az FCB-t, CX az (írandó) rekordszámot adja meg.

Ha nincs elég hely a lemezen, akkor az AL-ben 01-et ad vissza és írás nem történik. Ha AL-t 02-re állítja, a DTA volt túl kicsi és írás szintén nem történt. AL=00-t kapunk sikeres írás esetén.

Ha belépéskor a CX nullát tartalmaz, akkor írás nem történik, a file méretét viszont aktualizálja a tartalomjegyzékben a random rekord mezőbeli méret szerint.

Példa:

```
procedure RandomBlockWrite(fcb: fcbSet;
                           var counter: integer;
                           var mode: byte);
begin
   reg.DS:=Seg(fcb);
   reg.DX:=Ofs(fcb);
   reg.CX:=counter;
   reg.AH:=$28;
   MSDos(reg);
   mode:=reg.AL;
   counter:=reg.CX
end;
```

17.3.39 29-es funkció - file-azonosító keresése adott parancssorban

Belépéskor DS:SI az elemzendő parancs-sorra mutat, ES:DI pedig egy meg nem nyitott FCB-re. Az AL tartalmától függően kerül a művelet végrehajtásra a következőképpen:

Ha a 0. bit = 1, akkor a parancssor vezető elválasztó jelei a parancssorban figyelmen kívül maradnak (különben nem).

Ha az 1. bit = 0, akkor a parancssorból a meghajtó azonosítója átkerül az FCB-be, amennyiben van ilyen a parancssorban.

Ha 2. bit = 1, akkor a file-név a parancssorból átkerül az FCB-be, ha az tartalmaz file-nevet.

Ha 3. bit = 1, akkor ez ugyanaz, mint az előbbi, csak a file-név kiterjesztésére vonatkozik.

A 4-7 bitek tartalma közömbös.

A funkció outputja:

DS:SI a vizsgált file-azonosító utáni első karakterre mutat, ES:DI a megváltoztatott FCB-re mutat, AL tartalma pedig:

00, ha a parancssorban nem szerepeltek globális file-név karakterek,
01, ha szerepeltek,
FF, ha a meghajtó megadás rossz volt.

Példa:

```
procedure ParseFile(fcb: fcbSet;
                    var command: byte;
                    var hely: mutato);
begin
   reg.ES:=Seg(fcb);
   reg.DI:=Ofs(fcb);
   reg.DS:=Seg(hely|);
   reg.SI:=Ofs(hely|);
   reg.AL:=command;
   reg.AH:=$29;
   MSDos(reg);
   command:=reg.AL;
   hely:=ptr(rg.DS,reg.SI)
end;
```

17.3.40 2A funkció - dátum lekérdezése

AL binárisan a napot (0 vasárnap, 1 hétfő stb.), CX az évet (1980 és 2099 között), DH a hónapot (1 és 12 között), DL a napot (1 és 31 között) adja meg.

Példa:

```
procedure GetDate(var year,month,day,DayOfWeek: integer);
begin
   reg.AH:=$2a;
   MSDos(reg);
   year:=reg.CX;
   month:=reg.DH;
   day:=reg.DL;
   DayOfWeek:=reg.AL
end;
```

17.3.41 2B funkció - dátum beállítása

A funkció inputként CX-ben, DH-ban és DL-ben úgy várja a beállítandó dátumot, ahogy azt a 2A funkció leírásában megadtuk.

Outputként AL-ben 0-t ad, ha átírta a dátumot, FF-et, ha érvénytelen dátumot adtunk meg.

Példa:

```
procedure SetDate(year,month,day: integer;
                  var error:boolean);
begin
   reg.AH:=$2b;
   reg.CX:=year;
   reg.DH:=month;
   reg.DL:=day;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.42 2C funkció - rendszeridő lekérdezése

A funkció outputja:

CH az órát (0-23), CL a percet (0-59), DH a másodpercet (0-59) és DL a századmásodpercet (0-99) tartalmazza.

Példa:

```
procedure GetTime(var hour,minute,second,hundredth: integer);
begin
   reg.AH:=$2c;
   MSDos(reg);
   hour:=reg.CH;
   minute:=reg.CL;
   second:=reg.DH;
   hundredth:=reg.DL
end;
```

17.3.43 2D funkció - rendszeridő beállítása

A funkció inputját azokból a regiszterekből, és olyan formában várja, ahogy azt a 2C funkciónál leírtuk.

Ha az időmegadás érvénytelen, akkor az AL-ben FF-et, ha érvényes, akkor 00-t ad vissza (és átállítja a rendszeridőt).

Példa:

```
procedure SetTime(hour,minute,second,hundredth: integer
                  error: boolean);
begin
   reg.CH:=hour;
   reg.CL:=minute;
   reg.DH:=second;
   reg.DL:=hundredth;
   reg.AH:=$2d;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else error:=false
end;
```

17.3.44 2E funkció - VERIFY (lemezírás utáni ellenőrzés) kapcsoló beállítása

A funkció inputja:

AL-nek az ellenőrzés bekapcsolásakor 01-et, kikapcsolásakor 00-t kell tartalmaznia. Ha az ellenőrzés be van kapcsolva, akkor a lemezre történő íráskor ellenőrző visszaolvasást is végez a rendszer. Kritikus file-ok írásakor célszerű használni. A kapcsoló aktuális állása az 54-es funkcióval kérdezhető le.

Példa:

```
procedure SetVerify(state: byte);
begin
   reg.AL:=state;
   reg.AH:=$2e;
   MSDos(reg)
end;
```

17.3.45 2F funkció - lemez átviteli terület (DTA) címének lekérdezése

A funkció outputja:

ES:BX az aktuális lemez átviteli terület (DTA) címét tatalmazza.

Példa:

```
procedure GetDTA(cim: integer2);
begin
   reg.AH:=$2e;
   MSDos(reg);
   cim.HighInt:=reg.ES;
   cim.LowInt:=reg.BX
end;
```

17.3.46 30-as funkció - DOS verziószám lekérdezése

A funkció outputja:

AL a főverziószámot, AH a főverzión belüli változat értékét tartalmazza. (Például a 2.10 verzió esetén az AX-ben 0A02-t ad vissza a funkcióhívás.) A funkció PC-DOS verziónál nullázza a BX és CX regisztereket.

Megjegyzés:

Ha az AL 00-t tartalmaz, akkor ez azt jelenti, hogy egy, a DOS 2.00-t megelőző verziót használunk.

Példa:

```
function GetVersion: real;
begin
   reg.AH:=$2e;
   MSDos(reg);
   GetVersion:=reg.AL+reg.AH/100
end;
```

17.3.47 31-es funkció - programfutás befejezése úgy, hogy a program a memóriában marad

A funkció inputja:

AL egy bináris befejezési kódot, a DX a program számára lefoglalandó memóriaméretet tartalmazza paragrafusokban kifejezve. Ez a funkció befejezi az aktuális programot és a program kezdeti helyfoglalási igényét megpróbálja a DX-ben megadott számra átállítani.

Az AL-ben visszaadott befejezési kódot a programot hívó a WAIT funkcióhívással (4D), vagy (BATCH feldolgozásnál) az ERRORLEVEL alparanccsal ellenőrizheti.

Megjegyzés:

A befejezési kód felépítését lásd a 4D funkció ismertetésénél.

Példa:

```
procedure KeepProcess(ReturnCode: byte;
                      MemorySize: integer);
begin
   reg.AL:=ReturnCode;
   reg.DX:=MemorySize;
   reg.AH:=$31;
   MSDos(reg)
end;
```

17.3.48 32-es funkció - DOS számára fenntartott funkció

17.3.49 33-as funkció - Ctrl-Break lekérdezése és beállítása

A funkció inputja:

AL = 00 az aktuális állapot lekérdezését, AL = 01 az állapot beállítását jelenti. Ekkor DL = 00 a Ctrl-Break ellenőrzést kikapcsolja (OFF), DL = 01 pedig bekapcsolja (ON).

A funkció outputja:

Állapotlekérdezés esetén DL-ben jelzi vissza a ki-, illetve bekapcsolt állapotot, az előző értékek szerint.

Példa:

```
procedure CheckCtrlC(var SetBit, state: byte;
                     var error: boolean);
begin
   reg.AL:=SetBit;
   reg.DL:=state;
   reg.AH:=$33;
   MSDos(reg);
   if reg.AL=$ff then error:=true
   else
      begin
         error:=false;
         state:=reg.DL
      end
end;
```

17.3.50 34-es funkció - DOS számára fenntartott funkció

17.3.51 35-ös funkció - megszakítási (interrupt) vektor lekérdezése

Belépéskor az AL a megszakítás hexadecimális számát tartalmazza. A megadott megszakítást kezelő rutin címét a funkció ES:BX-ben adja vissza.

A rutin címét a 25-ös funkcióval lehet beállítani.

Példa:

```
procedure GetVector(number: byte;
                    cim: integer2);
begin
   reg.AL:=number;
   reg.AH:=$35;
   MSDos(reg);
   cim.HighInt:=reg.ES;
   cim.LowInt:=reg.BX
end;
```

17.3.52 36-os funkció - szabad lemezterület lekérdezése

Belépéskor DL a meghajtó sorszámát tartalmazza: 0=aktuális, 1=A stb. Ha a meghajtószám érvénytelen volt, akkor visszatéréskor az AX-be FFFF kerül, különben:

- BX a rendelkezésre álló lemezblokkok (cluster) számát,
- DX a meghajtón lévő összes lemezblokk számát,
- CX a szektoronkénti byte-ok számát,
- AX pedig clusterenkénti szektorok számát tartalmazza.

Megjegyzés:

DX, AL, CX ugyanezeket adja meg az 1B funkció hívása esetén is.

Példa:

```
procedure GetDiskSpace(drive: byte;
                        var FreeClusters,
                            BytesPerSector,
                            SectorsPerCluster,
                            ClustersPerDrive: integer;
                        var error: boolean);
begin
  reg.DL:=drive;
  reg.AH:=$36;
  MSDos(reg);
  if reg.AX:=$ffff then error:=true
  else
    begin
      FreeClusters:=reg.BX;
      BytesPerSector:=reg.CX;
      SectorsPerCluster:=reg.CX;
      ClustersPerDrive:=reg.AX;
      error:=false
    end
end;
```

17.3.53 37-es funkció - DOS számára fenntartott funkció

17.3.54 38-as funkció - ország-specifikus információk lekérdezése

Belépéskor DS:DX arra a 32 byte-os memóriaterületre mutat, ahová az ország specifikus információk kerülnek. AL vagy BX tartalmazza az ország kódját. Ha ez kisebb, mint 255, akkor AL, különben BX tartalmazza az ország kódját. Az utóbbi esetben AL értékét 255-re kell beállítani.(A DOS 2.00-ban AL-nek 00-nak kell lennie!)

A 32 byte-os puffer terület az alábbi infomációkat tartalmazza:

1 szó	A dátum és idő formáját jelző kód,
5 byte	ASCII-jelsorozat, a pénzegység jele, jobbról 0-val zárva,
2 byte	ASCII-jelsorozat, az ezresek elválasztásának jele, jobbról 0-val lezárva,
2 byte	ASCII-jelsorozat, a tizedes elválasztójel (pont, illetve vessző), jobbról 0-val lezárva,
2 byte	ASCII-jelsorozat, a dátum elválasztó jele, jobbról 0-val lezárva,
2 byte	ASCII-jelsorozat, az idő elválasztó jele, jobbról 0-val lezárva,
1 byte	Jelző bitek,
1 byte	Jelző: pénzegység használatakor a tizedes jegyek száma,
1 byte	Jelző: idő formátuma.
1 duplaszó	A kisbetűk/nagybetűk átalakító rutin kezdőcíme,
2 byte	ASCII-jelsorozat, az adatelválasztó jel jobbról 0-val lezárva,
10 byte	Fenntartott (a DOS használja).

A dátum és idő megadása formáját jelző szó értékei a következők lehetnek:

A szó értéke:	A szabványhoz tartozó dátum-megadás
0 = USA	szabvány: hónap nap év
1 = európai	szabvány: nap hónap év
2 = japán	szabvány: év hónap nap

A Jelzőbitek jelentése:

0. bit = 0 a pénzegység jele a szám előtt,
= 1 a pénzegység jele a szám után,
1. bit = 0 nincs szóköz a pénzegység és a szám között,
= 1 szóköz a pénzegység és a szám között.

Az idő formátum byte 0. bitjének leheséges értékei:

0 = 12 órás kijelzés,
1 = 24 órás kijelzés.

Példa:

```
procedure GetCountryData(CountryCode: integer;
                         var puffer: mutato;
                         var error: boolean);
begin
   if CountryCode<255 then reg.AL:=CountryCode
   else begin
      reg.AL:=255; reg.BX:=CountryCode
   end;
   reg.DS:=Seg(puffer|);
   reg.DX:=Ofs(puffer|);
   reg.AH:=$38;
   MSDos(reg);
   if (reg.Flags and 1)=1 then error:=true
   else error:=false
end;
```

17.3.55 39-es funkció - altartalomjegyzék létrehozása

Belépéskor DS:DX a meghajtót és elérési utat leíró ASCII jelsorozat címét tatalmazza.

Ha az elérési út valamelyik eleme nem létezik, akkor az altartalomjegyzék nem jön létre (Carry Yes), egyébként a megadott elérési út végére írja fel az új altartalomjegyzéket (No Carry).

Példa:

```
procedure MakeDir(pathname: MaxString;
                  var error: boolean);
```

```
begin
   pathname:=pathname+chr(0);
   reg.DS:=Seg(pathname);
   reg.DX:=Ofs(pathname);
   reg.AH:=$39;
   MSDos(reg);
   if (reg.Flags and 1)=1 then error:=true
   else error:=false
end;
```

17.3.56 3A funkció - altartalomjegyzék törlése

Belépéskor DS:DX egy olyan memóriaterületre mutat, ahol a meghajtót és elérési utat leíró ASCII-jelsorozat van.

Az így megadott altartalomjegyzék a struktúrából törlődik. Az aktuális tartalomjegyzék nem törölhető.

Ha a funkció végrehajtása után Carry = Yes, akkor az AX-beli érték a hibakódot jelenti.

Példa:

```
procedure DelDir(pathname: MaxString;
                 var error: boolean);
begin
   pathname:=pathname+chr(0);
   reg.DS:=Seg(pathname);
   reg.DX:=Ofs(pathname);
   reg.AH:=$3a;
   MSDos(reg);
   if (reg.Flags and 1)=1 then error:=true
   else error:=false
end;
```

17.3.57 3B funkció - aktuális alkönyvtár beállítása

Belépéskor DS:DX egy olyan memóriaterületre mutat, ahol a meghajtót és elérési utat leíró jelsorozat van.

Ha az elérési út valamely eleme nem létezik, nem történik változtatás (Carry Yes), különben az aktuális altartalomjegyzék a megadott lesz (No Carry).

Példa:

```
procedure SetDir(pathname: MaxString;
                 var error: boolean);
begin
   pathname:=pathname+chr(0);
   reg.DS:=Seg(pathname);
   reg.DX:=Ofs(pathname);
   reg.AH:=$39;
   MSDos(reg);
   if (reg.Flags and 1)=1 then error:=true
   else error:=false
end;
```

17.3.58 3C funkció - file létrehozása vagy meglévő file hosszának 0-ra állítása

Belépéskor DS:DX egy olyan memóriaterületre mutat, ahol a meghajtót, elérési utat és a file-azonosítót leíró ASCII-jelsorozat van, CX tartalmazza a file attributumot.

Ez a funkcióhívás egy új file-t állít elő vagy egy meglévő file-t 0-hosszúságúra állít. Ha a file nem létezik, akkor a megadott tartalomjegyzékbe bejegyzi. A file-hoz írási-olvasási elérési kódot rendel, és a file-t megnyitja. Ha a funkció rendben végrehajtódott, akkor a carry nincs beállítva, és AX-ben a file-számot nyerjük vissza.

Megjegyzés:

A file attributuma a 43-as funkcióhívással változtatható meg.

Példa:

```
procedure CreateFile(filename: MaxString;
                     attribute: byte;
                     var error: boolean);
begin
   fatname:=filename+chr(0);
   reg.DS:=Seg(filename);
   reg.DX:=Ofs(filename);
   reg.CX:=attribute;
   reg.AH:=$39;
   MSDos(reg);
   if (reg.Flags and 1)=1 then error:=true
   else error:=false
end;
```

17.3.59 3D funkció - file megnyitása

Belépéskor DS:DX egy komplett file-specifikációra mutat, AL az elérési kódot tartalmazza.

Az elérési kód megengedett értékei a DOS 2.11-3.00 változataiban:

0 = file megnyitása olvasásra,
1 = file megnyitása írásra,
2 = file megnyitása írásra/olvasásra.

A DOS 3.10-es verzióitól kezdve a kód byte bitjeinek a következő a jelentése:

7. bit:	öröklési bit,
6.-4. bitek:	megnyitási mód,
3.-0. bitek:	elérési mód.

Az öröklési bit jelzi, hogy a file átadandó-e egy, a processzor által elindított másik processzornak is. Ha a bit magas, az új processzor nem 'örökli' a file-t.

A megnyitási mód 4.-6. bitjeit a rendszer a következőképpen használja:

000	Kompatibilis mód. A file-t tetszőleges processzor megnyithatja, tetszőleges sokszor. Hibát jelez, ha bármilyen más módon próbálják megnyitni.
001	Teljes lefoglalás. A file-t ezután már semmilyen módon sem lehet megnyitni. Újra megnyitására még a megnyitó processzora sem képes. (Előtte le kell zárni a file-t)
010	Lefoglalás írásra. A file nem lesz megnyitva, ha egy másik processzor megnyitotta kompatibilis módban, vagy lefoglalta írásra.
011	Lefoglalás olvasásra. A file nem lesz megnyitva, ha egy másik processzor megnyitotta kompatibilis módban, vagy lefoglalta olvasásra.
100	Lefoglalás. A file nem lesz megnyitva, ha bármely más processzor megnyitotta kompatibilis módban.

Az elérési módok a következők:

0000	Olvasás. A file nem nyitható meg, ha már teljesen vagy olvasásra legfoglalták.
0001	Írás. A file nem nyitható meg, ha már teljesen vagy írásra legfoglalták.
0010	Mindkettő. A file nem nyiható meg, ha már teljesen, írásra vagy olvasásra lefoglalták.

Visszatéréskor az AX a hibakódot vagy a 16 bites file-számot tartalmazza aszerint, hogy a carry-t beállítva (hibakód) vagy nem beállítva (file-szám) kapjuk vissza.

A lehetséges hibák:

2 = a megadott file nem használható vagy nem létezik;
3 = a megadott út érvénytelen vagy nem létezik;
4 = a processzor számára nincs elérhető file-szám, vagy megtelt a DOS belső táblázata;
5 = a program egy altartalomjegyzéket, egy kötetazonosítót vagy egy csak-olvasható file-t próbált megnyitni;
12 = az elérési kód nem esik a 0-2 intervallumba.

A funkcióból való kilépés után az írás/olvasás a file első byte-jától kezdődhet, ide mutat az írás/olvasás pointer. Az aktuális rekordhossz 1 byte lesz. A pointer értékét a 42-es funkcióhívással módosíthatjuk, a rekordméretet olvasásnál a 3F, írásnál a 40-es funkciónál kell megadni.

A file utolsó megnyitásának dátumát és idejét az 57-es funkcióval állíthatjuk be vagy kérdezhetjük le, attributumát a 43-as funkcióval lehet lekérdezni, ill. módosítani.

Példa:

```
procedure Openfile(filename: MaxString;
                   mode: byte;
                   var filehandle: integer;
                   var error:integer);
begin
   filename:=filename+chr(0);
   reg.DS:=Seg(filename);
   reg.DX:=Ofs(filename);
   reg.AL:=mode;
   reg.AH:=$3D;
   MSDos(reg);
   if (reg.Flags and 1)=1 then error:=reg.AX
   else begin
      error:=0;
      filehandle:=reg.AX
   end
end;
```

17.3.60 3E funkció - file lezárása

A funkció a BX-ben megadott file-számú file-t zárja le (ezt a számot a 3C vagy 3D funkciók adták meg AX-ben).

A funkció a lezáráson kívül a tartalomjegyzék aktualizálását, és a file-hoz tartozó pufferek kiürítését is elvégzi. Hibajelzés: Carry bitben.

Példa:

```
procedure CloseFile(filehandle: integer;
                    var error:integer);
begin
   reg.BX:=filehandle;
   reg.AH:=$3E;
   MSDos(reg);
   if (reg.Flags and 1)=1 then error:=reg.AX
   else error:=0
end;
```

17.3.61 3F funkció - olvasás file-ból vagy egyéb perifériáról

A funkció a BX-ben megadott file-számú file-ból CX-ben megadott számú byte-ot olvas a DS:DX-ben megadott kezdőcímű pufferba.

Visszatéréskor a funkció AX-ben adja meg a ténylegesen olvasott byte-ok számát vagy sikertelen olvasás esetén (Carry Yes) a hibakódot.

Nem feltétlenül a CX-ben megadott számú byte-ot olvas a funkció, ugyanis például a klaviatúráról legfeljebb egy sornyi szöveg jöhet egyszerre, ill. ha a file végéig már kevesebb byte van, mint az adott szám, akkor annyit olvas, amennyi még rendelkezésre áll.

A standard inputról való olvasás átirányítható (lásd "A PC DOS Speciális lehetőségei" című 12. fejezetben a megfelelő szakaszt).

Ha AX-ben 0-t kapunk, a file végén túl kíséreltünk meg olvasni.

Példa:

```
procedure ReadHandle(handle:integer;
                     var count: integer;
                     var mode: integer);
begin
   reg.BX:=handle;
   reg.CX:=count;
   reg.DS:=Seg(puffer);
   reg.DX:=Ofs(puffer);
   reg.AH:=$3f;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else begin
      mode:=0;
      count:=reg.AX;
   end;
end;
```

17.3.62 40-es funkció - írás file-ba vagy egyéb perifériára

A funkció a BX-ben megadott file-számú file-ba a CX-ben megadott számú byte-ot ír a DS:DX-ben megadott című pufferből.

Akár e funkció olvasó párja, a 3F-es, ez is AX-ben adja vissza a ténylegesen írt byte-ok számát. Itt azonban ha (CX)=(AX), akkor hiba történt (például már nincs a lemezen hely), így programunkban ezt az esetet célszerű ellenőrizni.

Ha az írás még ennyire sem volt sikeres (Carry Yes), a hibakódot kapjuk AX-ben.

A standard output ugyanúgy átirányítható, mint a 3F funkció esetén az input.

17.3.63 41-es funkció - file törlése

Belépéskor DS:DX egy komplett file-specifikáció címét tartalmazza. A globális file-név karakterek használata nem megengedett.

A funkció a megadott file-t törli. Csak olvasható file-t a funkció nem tud törölni, ilyenkor a file attributumát - a funkció hívása előtt - a 43-as funkcióval 0-ra kell állítani.

Példa:

```
procedure DeleteDirectory entry(path:string64;
                                var mode:integer);
begin
   path:=path+chr(0);
   reg.DS:=Seg(path);
   reg.DX:=Ofs(path);
   reg.AH:=$41;
   MSDos(reg);
   if (reg.Flags and 1)=0 then mode:=0
   else mode:=reg.AX
end;
```

17.3.64 42-es funkció - írási/olvasási mutató átállítása

Amikor egy file-t a 3D funkcióval megnyitunk, az írási/olvasási mutató a file első byte-jára mutat.

A file feldolgozása során ezt a mutatót mindig be kell állítani, ha nem sorosan olvasunk/írunk.

Belépéskor BX tartalmazza a kezelendő file számát, CX:DX azt az értéket, hogy a mutató kurrens értéke az AL-ben meghatározott módon számítva hány byte-tal módosuljon - ez az érték az ún. eltolási vagy offset érték. (CX tartalmazza a magasabb helyiértékű részt.)

Az eltolást AL a következőképpen határozza meg:

0 = a file elejétől,
1 = a mutató jelenlegi állásától,
2 = a file végétől visszafelé.

A funkció output-ként DX:AX-ben adja meg a mutató új helyzetét, ha a carry nem volt beállítva. (DX-ben van a magasabb helyiértékű rész.)

Példa:

```
procedure MovePointer(handle:integer;
                      var distance: integer2;
                      method: byte;
                      var mode: integer);
begin
    reg.AL:=method;
    reg.BX:=handle;
    reg.CX:=distance.High;
```

```
   reg.DX:=distance.Low;
   reg.AH:=$42;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=1
   else begin
      mode:=0;
      distance.High:=reg.DX;
      distance.Low:=reg.AX;
   end;
end;
```

17.3.65 43-as funkció - file attributumának lekérdezése vagy módosítása

Ha belépéskor (AL) = 0, a funkció lekérdezi, ha (AL) = 01, módosítja a file attributumát a következők szerint:

DS:DX egy komplett file-specifikáció címét tartalmazza, CX-ben adjuk, illetve kapjuk meg a file attributumát.

Megjegyzés:

A hexadecimális 08 és 10 attributum (kötetazonosító, illetve altartalomjegyzék jelzés) nem változtatható meg.

Példa:

```
procedure FileAttribute(file name:string64;
                        command: byte;
                        var attributes: byte;
                        var mode: integer);
begin
   file name+chr(0);
   reg.DS:=Seg(file name);
   reg.DX:=Ofs(file name);
   if command=1 then reg.CX:=attributes;
   reg.AL:=command;
   reg.AH:=$43;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else begin
      mode:=0;
      if command=0 then attributes:=reg.CX;
   end;
end;
```

17.3.66 44-es funkció - periféria vagy file-leíró információ lekérdezése vagy módosítása

A funkció inputja:

Az AL regiszter tartalmával határozzuk meg, hogy a sok lehetőség közül a funkció melyik feladatot lássa el.

DS:DX egy adat vagy puffer címe.
CX az írandó vagy olvasandó byte-ok száma (AL = 02,03,04 vagy 05 alfunkció esetén).
BX a file-szám (AL = 00,06,07,0A,0C esetén).
BL a meghajtó azonosítója (AL = 04,05,08,09,0C,0E vagy 0F esetén).

A DOS eltérő verzióiban nem mindegyik alfunkció használható:

DOS -2.11:	00H-07H
DOS 3.0 és 3.1:	00H-0BH
DOS 3.2-:	00H-0FH

A funkció a következő feladatok ellátására képes:

AL = 00	periféria leíró információ lekérdezése DX-be - az információ részletezése majd következik,
AL = 01	periféria leíró információ átállítása, amelyet DX-ben adunk meg. DL-ben a táblázat 7-0. bitje módosítható.),
AL = 02	(CX) számú byte olvasása DS:DX-be a periféria vezérlő csatornájáról. (BX) adja meg a perifériát,
AL = 03	ugyanennek az írása, lásd AL = 02,
AL = 04	ugyanaz, mint 02, csak nem a file-számmal kijelölt perifériára, hanem a (BL)-ben a meghajtó-azonosítóval adjuk meg a perifériát,
AL = 05	a 03 megfelelője, írás (BL) szerint. A 02,03,04 és 05-ös értékekkel tetszőleges vezérlő karaktersorozatok küldhetők vagy fogadhatók.
AL = 06	az inputra, és
AL = 07	az outputra való felkészültség állapotát kérdezi le, a következő módon: Ha file-számot adunk meg a 06 és 07-es értékekhez, akkor AL-ben mindaddig FF-et kapunk vissza, míg a file végét el nem érjük, ezután pedig 00-t, hacsak át nem állítjuk a 42-es funkcióval az írási/olvasási mutatót. Ha perifériát adunk meg, FF-et kapunk AL-ben készenléti, és 00-t nem készenléti állapot esetén.
AL = 08	azt kérdezi meg, hogy a BL-ben megadott egység cserélhető vagy sem. Ha igen, akkor a visszatéréskor AX = 0, különben 1.

AL = 09 ellenőrzi, hogy a BL-ben megadott egység lokális, vagy hálózaton implementált. Ha az egység lokális, akkor a rutin DX-ben az egység perifériakezelőjének állapotszavát adja vissza. Ha nem, akkor a DX-nek a 12. bitje magas, a többi 0.

AL = 0A ellenőrzi, hogy a BX-ben megadott file-számot kezelő periféria handler hálózaton implementált vagy lokális. Ez utóbbi esetben a visszatéréskor a DX-ben az IOCTL adatok találhatók. Ha a file-szám nem lokális, akkor a DX 15.bitjét a rutin magasra állítja.

AL = 0B alfunkcióval beállíthatjuk, hogy a DOS hányszor próbálkozzon egy lemezművelettel, ha az file zárolása miatt nem hajtható végre. DX-ben kell megadni a probálkozások számát. Ha a visszatéréskor a Carry magas (= 1), akkor a file megosztási környezetet nem töltöttük be a SHARE DOS paranccsal, s ezért az alfunkció nem hajtható végre. Ilyenkor AX értéke 1.

AL = 0C segítségével a nyomtatás-míg-foglalt iterációszáma kérdezhető le. BX-ben kell a file-számot megadni. A CH értékét 05H-ra kell beállítani. Ha CL = 45H, akkor az alfunkció beállítja, ha CL = 65H, akkor lekérdezi a számlálót. A meghíváskor a DS:DX egy szóra mutat, amelyik a számlálót tartalmazza. CL értékétől függően ezt is fel kell tölteni, vagy visszatéréskor innen olvasható ki a kért információ.

AL = 0D segítségével a blokkos berendezések bizonyos adatai is beállíthatók. Megint BL-ben kell megadni az egységet. A CH értéke kötelezően 08H. A CL értéke határozza meg az elvégzendő feladatot:

- 40 = paraméterek beállítása;
- 60 = paraméterek lekérdezése;
- 41 = sáv írása a logikai egységen;
- 61 = sáv olvasása a logikai egységről;
- 42 = sáv formázása a logikai egységen;
- 62 = sáv ellenőrzése a logikai egységen;

DS:DX minden esetben az adott funkcióhoz tartozó puffer elé mutat (= puffer kezdőcíme - 1). Ezt a CL értéktől függően nekünk kell állítani, vagy onnan kapunk információt.

AL = 0E lekérdezi a logikai-fizikai egységmegfeleltetést. BX tartalmazza az egység betűjelét (0 = aktuális, 1 = A stb.) A hibát a Carry magasra állítása (= 1) jelzi. Ha nem, akkor AL annak a logikai egységnek a betűjele, amelyik utoljára használta a szóbanforgó fizikai egységet. Ez 0, ha a fizikai egységhez csak egy logikai egység tartozik.

AL = 0F beállítja a logikai-fizikai egységmegfeleltetést. BX tartalmazza az egység betűjelét (0 = aktuális, 1 = A stb.). Az alfunkció hatására a fizikai egységet a soronkövetkező logikai egységhez rendeli hozzá a rendszer. A hibát a Carry magasra állítása (= 1) jelzi. Ha nem, akkor AL annak a logikai egységnek a betűjele, amelyik újonnan használja a fizikai egységet.

Megjegyzés:

Valamely file-szám akkor kezel perifériát, ha a megnyitáskor (3D funkció) megadott filenév a DOS számára periféria-név (pl. standard PRN), vagy a CONFIG.SYS-ben kijelölt valamelyik felhasználói periféria-kezelő rutin tartalmazza a nevet.

A 44-es funkció outputja:

A 02,03,04 és 05-ös (AL) esetén az átvitt byte-ok számát, a 06 és 07 esetén a készenléti vagy nem készenléti állapotot kapjuk hibátlan működés esetén AX-ben.

Hiba esetén a Carry magas lesz(= 1).

Az AL = 0 esetén a funkciónak DX-ben is van outputja.

Ha AL = 0 vagy 1, DX bitjei a következő információt tartalmazzák:

15	14	13...8	7	6	5	4	3	2	1	0
fenntartott	C	fenntartott	I	E	B	fenntartott	I	I	I	I
	T		S	O	I		S	S	S	S
	R		D	F	N		C	N	C	C
	L		E				L	U	O	I
			V				K	L	T	N

ahol:

ISDEV = 1 karakteres berendezés esetén.
= 0 blokkos berendezés (lemez-file) esetén (ekkor a 8-15 bitek 0 értékűek).
ISDEV = 1 esetén:
EOF = 0, ha input esetén EOF-rekordot olvasott,
BIN = 1, ha bináris módon történik a feldolgozás (a CTRL-Z-t nem ellenőrzi),
= 0, ha ASCII módon történik (a CTRL-Z-t EOF-ként értelmezi),
ISCLK = 1, ha a berendezés óra (clock device),
ISNUL = 1, ha ez null-egység (null device).

A null-egység egy nem létező berendezés. Használhatjuk DOS parancsban is, de nem küld és nem fogad információt. Például, ha egy COPY parancsnak ez az outputja, a COPY a forrásról olvas, de nem ír sehova - ezt a lehetőséget ellenőrzésre lehet használni.

ISCOT = 1, ha ez a periféria a konzol output.
ISCIN = 1, ha ez a periféria a konzol input.
CTRL = 0, ha az AL = 2,3,4,5 értékek erre a perifériára nem alkalmazhatók.
CTRL = 1, ha a 2,3 műveleti kód is kiadható. Ezt a bitet beállítani nem lehet a funkcióval.

ISDEV = 0 esetén:
EOF = 0, ha az adatbevitel befejeződött.

A 0-5 bitek blokkos periféria esetén a periféria azonosítóját (0 = A, 1 = B stb.) adják meg.

A 15, 8-13 és 4-es bitek értékét a felhasználó nem módosíthatja.

17.3.67 45-ös funkció - egy file file-számának duplikálása

A funkció BX-ben egy nyitott file file-számát várja.

AX-ben visszaad egy másik file-számot, amely ugyanarra a file-ra és ugyanarra a pozícióra (mutató-állásra) mutat, mint az eredeti file. Az új file attributum byte-ja ugyanaz, mint a régié. A továbbiakban ugyanannak a file-nak két file-száma lesz.

Példa:

```
procedure DuplicateHandle(handle: integer;
                          var handle2: integer;
                          mode: integer);
begin
   reg.BX:=handle;
   reg.AH:=$45;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else begin
     mode:=0;
     handle2:=reg.AX
   end
end;
```

17.3.68 46-os funkció - file-szám duplikálás úgy, hogy a másodpéldányt a felhasználó adja meg

A funkció BX-ben egy létező file file-számot, CX-ben egy másik számot vár.

Ha CX egy nyitott file-számot tartalmaz, ezt a file-t a funkció lezárja, majd a CX-beli file-számot átirányítja úgy, hogy az ugyanarra a file-ra és annak ugyanarra a pozíciójára mutasson a további file-kezdeti utasítások szempontjából, mint a BX-beli file-szám.

Ezután, ha a két file-szám bármelyikét felhasználva, a file írási/olvasási mutatóját átállítjuk, a másik file-szám pointere ugyanúgy változik.

Példa:

```
procedure ForceDuplicateHandle(handle: integer;
                          handle2: integer;
                          mode: integer);

begin
   reg.BX:=handle;
   reg.CX:=handle2;
   reg.AH:=$46;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else mode:=0;
   handle2:=reg.AX;
end;
```

17.3.69 47-es funkció - adott meghajtó aktuális alkönyvtárának lekérdezése

A funkció DS:SI-ben egy 64 byte-os felhasználói memóriaterület címét , DL-ben a meghajtó azonosítóját (0=aktuális, 1=A stb.) várja. Erre a memóriaterületre teszi a gyökér tartalomjegyzéktől részletesen kifejtett elérési utat. (A meghajtó betűazonosítója ebben nem szerepel). Az utat leíró karaktersor nem kezdődik backslash-sel(\). A string végét egy 0 byte jelzi.

Példa:

```
procedure GetDirectory(var path:string64
                           drive: byte;
                           var mode:integer);
begin
   reg.DS:=Seg(path);
   reg.DX:=Ofs(path);
   reg.DL:=drive;
   reg.AH:=$47;
   MSDos(reg);
   if (reg.Flags and 1)=0 then mode:=0
   else mode:=reg.AX
end;
```

17.3.70 48-as funkció - memóriafoglalás

A funkció BX-ben várja a kívánt memóriméretet, paragrafusban kifejezve. (Lásd: a DOS munkaterületei című fejezet Memória nyilvántartás című részét.)

A funkcióból való visszatéréskor AX:0 mutat a lefoglalt memória blokkra. Ha a kért mennyiség nem áll rendelkezésre, a memóriából még lefoglalható mennyiséget BX-ben kapjuk paragrafusokban kifejezve.

Példa:

```
procedure AllocateMemory(var memory: integer;
                         var address: integer;
                         var mode: integer);

begin
   reg.BX:=memory;
   reg.AH:=$48;
   MSDos(reg);
   if (reg.Flags and 1)=1 then begin
     mode:=AX;
     memory:=reg.BX
   end
     else begin
     mode:=0;
     address:=reg.AX
   end
end;
```

17.3.71 49-es funkció - lefoglalt memóri felszabadítása

A funkciónak ES-ben adjuk meg a visszaadandó memóriablokk szegmenscímét. (A memóriablokkok szegmenshatáron kezdődnek.) Ezt a funkció a rendszernek továbbítja, azaz ez a memóriablokk felszabadul. (A DOS minden memóriablokk elején felépít egy vezérlő blokkot és ebben tartja nyilván, hogy az illető blokk, illetve annak egy része foglalt-e vagy szabad.)

Példa:

```
procedure FreeMemory(page: integer;
                       var mode: integer);
begin
   reg.ES:=page;
   reg.AH:=$49;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
      else mode:=0;
   handle2:=reg.AX
end;
```

17.3.72 4A funkció - lefoglalt memóriablokk méretének megváltoztatása

A funkciónak ES-ben adjuk meg a módosítandó memóriablokk szegmenscímét, BX-ben a kívánt új blokkméretet paragrafusban kifejezve.

A funkció hívásakor a DOS megpróbálja a megadott blokk méretét a megadott méretre növelni vagy csökkenteni. Ha a méret növelése nem sikerül, BX-ben adja vissza a rendelkezésre álló maximális blokkméretet. Ilyenkor Carry-t a rendszer magasra állítja (= 1).

Példa:

```
procedure ModifyMemory(page: integer;
                        var size: integer;
```

```
                               var mode: integer);
begin
   reg.BX:=size;
   reg.ES:=page;
   reg.AH:=$4A;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else begin
     mode:=0;
     size:=reg.BX
   end
end;
```

17.3.73 4B funkció - program betöltése végrehajtatással vagy anélkül

A funkció lehetővé teszi, hogy a felhasználói program egy másik programot betöltsön, akár végre is hajtasson.

DS:DX-ben a funkció egy olyan ASCII sztring címét várja, amely tartalmazza a betöltendő program file-specifikációját. ES:BX a betöltéshez szükséges paraméter-blokkra mutat. A paraméter-blokk tartalmát AL értéke szabja meg. Hasonlóan az AL értéke határozza meg, hogy a DOS elindítja-e a betöltött programot:

AL = 0 esetben a DOS létrehozza a program számára a PSP-t, a befejezés és a Ctrl-Break esetén felhasználandó kilépési címeket pedig a hívó programba irányítja a funkcióhívás utáni helyre, ezután pedig elindítja a betöltött programot.

AL = 3 esetében a DOS nem hoz létre PSP-t, és nem indítja el, csak betölti a programot.

Megjegyzés:

AL = 0 esetén, amikor a főprogram visszakapja a vezérlést, az összes regiszter és a verem (stack) tartalma is változik, így az SS, SP és a többi szükséges regisztert a funkció hívása előtt ki kell menteni. Amikor a hívott program befejezte a működését, vissza lehet tölteni.

A paraméter-blokk tartalma a következő:

a) *AL = 00*

1. szó A betöltendő program programkörnyezetét megadó (legfeljebb 32K hosszú) sztring szegmenscíme. Ha ez 0, akkor a betöltendő program örökli a hívó program programkörnyezetét. A környezet szegmenscíme, amint az a PSP leírásában is látható, a betöltendő program PSP-jének 2C helyére (PSP + 2CH) kerül.

A fent említett sztringben megadható lehetőségeket szintén a PSP leírásánál ismertetjük. A környezetleíró sztring paragrafushatáron kezdődik.

2.-3.szó Arra a parancsra mutat, amelyet a rendszer a betöltendő program PSP + 80 helyére tesz.

4.-5.szó A betöltendő program PSP + 5C helyére teendő FCB-re mutat.

6.-7.szó A PSP + 6C-re teendő FCB-re mutat.

b) *AL = 03*

1. szó Az a szegmenscím, ahová a file-t be lehet tölteni.

2. szó Az az eltolási(relocation) érték, aminek figyelembe vételével a programot be kell tölteni (lásd az .EXE file szerkezetének ismertetését).

Megjegyzés:

Az összes olyan file, amelyet a főprogram megnyitott, a betöltendő program számára is nyitva van, elérhető. Így egyes részfeldolgozási feladatokat kényelmesen struktúrálhatunk.

A 4B funkció a betöltéshez a COMMAND.COM-ban lévő betöltő programot használja. Ha a felhasználói program a betöltő programot átfedte, akkor ezzel a funkcióhívással újból a memóriába kerül. Ha a felhasználó a teljes memóriaterületet igényelte, akkor a funkcióhívás hibaüzenetet ad, mivel nincs hely a betöltő program új példánya számára.

17.3.74 4C funkció - program befejezése

A funkció hívásakor - opcionálisan - AL-be egy bináris számot, úgynevezett befejezési kódot tehetünk. A funkció az éppen futó programot befejezi, azokat a file-okat, amelyeket megnyitott, lezárja és a vezérlést a parancsfeldolgozónak, illetve a hívó programnak adja át. A befejezési kódot az IF alparancs IF ERRORLEVEL formájával, vagy a 4D funkcióval le lehet kérdezni. Más outputja nincs a funkciónak.

Példa (a 4B és 4C funkciókhoz):

```
procedure LoadProgram (ProgramName:szoveg;
                       ParameterBlock:szoveg;
                       OverlayFlag:boolean;
                       var mode: integer);
begin
   reg.DS:=Seg(ProgramName);
   reg.DX:=Ofs(ProgramName);
   reg.ES:=Seg(ParameterBlock);
   reg.BX:=Ofs(ParameterBlock);
   if overlayFlag then reg.AL=$03 else reg.AL=0;
   reg.AH:=$4B;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else mode:0
end;

procedure EndProcess(ReturnCode: byte);
begin
   reg.AL:=ReturnCode;
   reg.AH:=$4C;
   MSDos(reg)
end;

function GetReturnCode:integer;
begin
   reg.AH=$4D;
   MSDos(reg);
   GetReturnCode:=reg.AX
end;
```

17.3.75 4D funkció - a hívott program befejezési kódjának lekérdezése

A hívó program ezzel a funkcióval kaphatja meg - az AL regiszterben - azt a befejezési kódot, amelyet a 31-es vagy 4C funkcióval a hívott program üzen.

AH értéke a következő lehet:

- 00 = normál befejezés;
- 01 = CTRL-BREAK befejezés;
- 02 = befejezés periféria hiba miatt;
- 03 = befejezés 31-es funkcióval

Megjegyzés:

Amikor a kódot lekérdezzük, dolgozzuk is azonnal fel, mert a 4D funkció csak egyszer adja vissza. Az ismételt hívás eredménytelen lesz.

17.3.76 4E funkció - elérési úttal megadott file keresése az első találatig

A funkció DS:DX-ben egy olyan ASCII sztring címét várja, amely a keresett file meghajtóját, elérési útját és nevét tartalmazza (szerepelhetnek globális file-név karakterek is). CX-ben az attributum byte értékét kell megadni, ami megmutatja, hogy milyen file-okat keresünk.

A funkció a DTA-ba adja vissza az először megtalált egyező file információit a következő módon:

21 byte	fenntartva a 4E funkciót esetleg követő 4F funkcióhívás számára;
1 byte	a megtalált file attributuma;
2 byte	létrehozásának ideje;
2 byte	létrehozásának dátuma;
2 byte	méretének alacsonyabb helyiértékű szava;
2 byte	méretének magasabb helyiértékű szava;
12 byte	a file neve és kiterjesztése (szóközök nélkül és az esetleges kiterjesztés előtt pont van, azaz olyan formátumban, ahogy egy parancsban paraméter lehet),
1 byte	0 tartalmú.

Példa:

```
procedure FindFirstFile (FileName: szoveg;
                         Attributes: integer;
                         var mode: integer);
begin
   reg.DS:=Seg(FileName);
   reg.DX:=Ofs(FileName);
   reg.CX:=Attributes;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else mode:=0;
     handle2:=reg.AX
   end
end;
```

17.3.77 4F funkció - további, a megadottnak megfelelő file keresése (4E hívás után)

A funkció inputja ugyanaz, mint a 4F funkcióé, feladata pedig a 4E hívása után további megfelelő file-ok keresése. Ha a 4F további megfelelő file-okat talál, a DTA-t úgy tölti fel, mint 4E-nél, egyébként hibakódot ad. Ez akkor is megtörténik, ha nem talál megfelelő file-t.

Példa:

```
procedure FindNextFile (mode: integer);
begin
   reg.AH:=$4F;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else mode:=0
end;
```

17.3.78 50-53-as funkció - DOS számára fenntartott funkció

17.3.79 54-es funkció - VERIFY kapcsoló értékének lekérdezése

A VERIFY kapcsoló értéke határozza meg, hogy végezzen-e a rendszer ellenőrzést írás után (lásd az ugyanilyen nevű DOS parancsot).

Az 54-es funkció segítségével hozhatjuk be ezt az értéket programunkba.

A funkció inputot nem kíván.

Végrehajtása után AL értéke adja a kapcsoló értékét:

AL = 00, ha az érték OFF és
AL = 01, ha ON.

Megjegyzés:

A 2E funkcióval állíthatjuk be a kapcsoló értékét.

Példa:

```
function GetVerifyState: byte;
begin
   reg.AH:=$54;
   MSDos(reg);
   GetVerifyState:= reg.AL
end;
```

17.3.80 55-ös funkció - DOS számára fenntartott funkció

17.3.81 56-os funkció - file átnevezése

A funkció DS:DX-ben egy komplett file-specifikációt tartalmaz. A névben nem szerepelhetnek globális file-név karakterek. ES:DI-ben pedig azon ASCII string címét kell megadni, amely az új nevet a következőképpen adja meg: Ha szerepel meghajtó, az ugyanaz, mint az első string-ben. Az elérési út és a file-név természetesen más is lehet. A file-név a második string-ben sem tartalmazhat globális file-név karaktereket.

Példa:

```
procedure RenameFile (FileNameOld,FileNameNew:szoveg;
                      var mode: integer);
begin
   reg.DS:=Seg(FileNameOld);
   reg.DX:=Ofs(FileNameOld);
   reg.ES:=Seg(FileNameNew);
   reg.DI:=Ofs(FileNameNew);
   reg.AH:=$56;
   MSDos(reg);
if (regFlags and 1)=1 then mode:=reg.AX
   else mode:=0
end;
```

17.3.82 57-es funkció - file dátumának és időpontjának lekérdezése, illetve módosítása

Az, hogy a funkció lekérdezést vagy beállítást végez-e, AL tartalmától függ:

AL = 00 lekérdezés,
AL = 01 beállítás.

BX-ben adjuk meg a file-számot és AL = 01 esetén CX-ben a beállítandó időpontot, DX-ben pedig a beállítandó dátumot. Ez utóbbi két adatot ugyanabban az alakban kell megadni, ahogy ezek a tartalomjegyzék bejegyzésben szerepelnek, csak a byte-ok sorrendjét az egyes regiszterekben meg kell fordítani, azaz pl. DH-ban kell a dátum alacsonyabb helyiértékű részét megadni.

Ha a funkció feladata a beállítás volt, az esetleges hibakódon kívül nincs outputja. Ha lekérdezés volt, akkor CX-ben kapjuk a file létrehozásának, illetve utolsó módosításának időpontját, DX-ben a dátumát.

17.3.83 58-as funkció - memória helyfoglalás módjának lekérdezése, illetve beállítása

A funkció inputjaként AL-ben 0-t kell megadni, ha lekérdezni, 1-et, ha beállítani akarjuk a memória helyfoglalás módját. Ez utóbbi esetben BX-ben a választott helyfoglalás kódját kell megadni. Lekérdezéskor ezt a kódot AX-ben kapjuk vissza.

A funkció csak a DOS 3.20-tól használható!

A lehetséges kódok és jelentésük:

0 **Első találat.** A DOS az első - a kérésnek megfelelő memóriablokkot foglalja le. Ez a kezdeti (default) mód is.
1 **Legjobb találat.** A DOS végigvizsgálja az összes lehetséges memóriablokkot, s a kérésnek megfelelők közül a legkisebbet foglalja le.
2 **Utolsó találat.** A DOS a legmagasabb helyen található blokkoktól lefelé kezd el keresni, s az első megfelelő blokkot foglalja le.

Ha hiba történik (AL vagy BX tartalma nem megfelelő), akkor a DOS a Carry flag-et magasra (= 1) állítja.

Példa:

```
procedure AllocationStrategy(code: byte;
                             var method: integer;
                             var mode: integer);

begin
   reg.AH:=$58;
   if code=1 then reg.BX:=method;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else
   if code=0 then method:=reg.AX
end;
```

17.3.84 59-es funkció - a hiba leírásának lekérdezése

Ennek a funkciónak a segítségével a hiba okáról további információkat kaphatunk. A felhasználó által készitett 24H megszakítás kezelők pl. használhatják ezt a funkciót a hibák elemzésére. A funkció meghívásakor a BX-be 0-t kell helyezni, ez a rendszer továbbfejlesztésekor majd jelzi, hogy milyen verziójú hibakezelő rendszert használunk.

A funkció csak a DOS 3.00-tól használható!

A funkció outputja:

BH tartalmazza a hiba osztályát. Az egyes értékek jelentése:

1 = Nem megengedett paraméter (pl. memória, csatorna-szám)
2 = Ideiglenes probléma (pl. file zárolása)
3 = Hozzáférési probléma
4 = Software rendszer belső hibája
5 = Hardware hiba
6 = Software rendszer hiba, amit nem a processz maga okozott (pl. nem megfelelő CONFIG.SYS file)
7 = Alkalmazói program hiba
8 = A file vagy valami egyéb nem található
9 = A file vagy valami egyéb nem megfelelő formátumú
10 = Zárolt file vagy valami egyéb tétel
11 = Rossz lemez vagy egyéb tárolási eszköz probléma
12 = Egyéb hiba
13 = Ismeretlen hiba

BL tartalmazza a javasolt hibajavítási mód kódját. Az egyes kódok jelentése a következő:

1 = Próbálkozzon újra, majd kérdezze meg a felhasználót.
2 = Próbálkozzon egy kis szünet után.
3 = Ha a felhasználó által megadott adat volt, kérdezze meg újra.
4 = Fejezze be a karbantartási funkciók elvégzése után.
5 = Azonnal be kell fejezni!
6 = A hiba csak informális jellegű, nincs igazi hiba.
7 = Kérdezze meg a felhasználót néhány konkrét dologról, majd ismételje meg (pl. cserélje ki a lemezt).

A CH **a hiba helyére utaló** információt tartalmaz:

1 = ismeretlen;
2 = véletlen elérésű eszköz (pl. lemezegység);

3 = hálózat;
4 = soros elérésű eszköz;
5 = a memória.

Példa:

```
procedure GetError(var ErrorCode:integer;
                   var ErrorClass,Action,Locus: byte);
begin
   reg.AH:=$59;
   reg.BX:=0;
   MSDos(reg);
   ErrorCode:=reg.AX;
   ErrorClass:=reg.BH;
   Action:=reg.BL;
   Locus:=reg.CL;
end;
```

17.3.85 5A funkció - file létrehozása egyedi névvel

A funkció egy olyan file-t hoz létre, amilyen nevű file még nincs a rendszerben. A funkció meghívásakor DS:DX egy út nevet leíró sztringre mutat, amelyik 0 byte-tal végződik. Ennek az útnak megfelelő szótárban fog a funkció egy új file-t létrehozni. CX a létrehozandó file attributumait tartalmazza.

A funkció csak a DOS 3.00-tól használható!

A visszatéréskor AX a megnyitott file-hoz rendelt file-számot tartalmazza, míg a funkció a DS:DX sztringhez hozzáírja a file nevét. Így az egyéb műveletekben használható.

A hívó processzor törlése nem törli az ideiglenes file-t, arról magunknak kell gondoskodni.

Ha hiba történik, akkor a Carry flag magas (= 1) lesz, s AX tartalmazza a hiba kódját:

2 = A file attribútum nem megfelelő
3 = A megadott útleírás nem megfelelő
4 = Túl sok a megnyitott file
5 = Az elérés nem megengedett

Példa:

```
procedure CreateUniqueFile (var PathName:szoveg;
                            Attributes: integer;
                            var handle: integer;
                            var mode: integer);

begin
   reg.DS:=Seg(PathName);
   reg.DX:=Ofs(PathName);
   reg.CX:=Attributes;
   reg.AH:=$5A;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else begin
     mode:=0;
     handle:=reg.AX
   end
end;
```

17.3.86 5B funkció - új file létrehozása

A funkció segítségével egy még nem létező file-t hozhatunk létre. A meghíváskor DS:DX a file nevére mutat (végén egy 0 byte-tal), és CX a létrehozandó file attribútumait tartalmazza. Ha a file még nem létezik, akkor a funkció létrehozza, megnyitja kompatibilis módban, s egy file számot rendel hozzá. Ha létezik - akkor szemben a 3C funkcióval - nem csonkolja 0 hosszúságra, hanem hibajelzést ad.

Visszatéréskor csak hibaüzenetet kapunk. A Carry flag magas (= 1) volta jelzi a hibát. AX a hiba okának kódját tartalmazza:

2 = a file-megadás hibás
3 = a megadott útleírás nem megfelelő
4 = az elérés nem megengedett
80 = a file már létezik

Megjegyzés:

A funkció kiválóan alkalmas szemafornak (test and set) hálózaton együttműködő programoknál.

Példa:

```
procedure CreateNewFile (var PathName:szoveg;
                         Attributes: integer;
                         var Handle: integer;
                         var Error: boolean);
begin
   reg.DS:=Seg(PathName);
   reg.DX:=Ofs(PathName);
   reg.CX:=Attributes;
   reg.AH:=$5B;
   MSDos(reg);
   if (reg.Flags and 1)=1 then Error:=true
   else begin
     Error:=false;
     Handle:=reg.AX
   end
end;
```

17.3.87 5C funkció - file-zárolás

A funkció két alfunkcióra tagolódik: 00 = zárolás és 01 = zárolás feloldása. A zárolással megtiltható, hogy a megadott file kijelölt területéhez más program hozzáférjen. A feloldással a kijelölt területet ismét elérhetővé tehetjük más programok számára is. E funkciót kell használni hálózati alkalmazásoknál, ha ugyanazon file-okon egyidőben több munkahelyről is akarnak dolgozni. A funkciókódot (00 vagy 01) az AL regiszterben, a file-számot a BX-ben, a zárolni kívánt file-terület kezdetét - a byte helyét a file elejéhez viszonyítva - a CX:DX regiszterpárban, a zárolandó szakasz hosszát - byte-ban - az SI:DI regiszterpárban kell megadni.

Visszatéréskor csak hibaüzenetet kapunk. A Carry flag magas (= 1) volta jelzi a hibát. AX a hiba okának kódját tartalmazza:

1 = érvénytelen alfunkciókód
6 = érvénytelen file-szám
33 = zárolási problémát
36 = a megosztást tároló puffer túlcsordult

Figyelem! Ha egy olyan file-t zárunk le, amelyben zárolt részek vannak, akkor a file tönkremehet! Lezárás előtt ezért valamennyi zárolást oldjunk fel, s csak utána zárjuk le a file-t.

Példa:

```
procedure Locking(Code: byte;handle: integer;
                  FirstByte, LengthOfRegion: integer2;
                  var mode: integer);
begin
   reg.AL:=Code;
   reg.BX:=handle;
   reg.CX:=FirstByte.High;
   reg.DX:=FirstByte.Low;
   reg.SI:=LengthOfRegion.High;
   reg.DI:=LengthOfRegion.Low;
   reg.AH:=$5C;
   MSDos(reg);
   if (reg.Flags and 1)=1 then mode:=reg.AX
   else mode:=0
end;
```

17.3.88 5E funkció - hálózati funkciók

Az 5E funkciók a hálózatkezeléssel kapcsolatos DOS funkciók. Az egyes alfunkciók - aminek kódját AL-ba kell elhelyezni - jelentése teljesen eltér, ezért külön ismertetjük ezeket.

A funkció csak a DOS 3.10-től használható!

00-ás alfunkció - a lokális gép nevének lekérdezése

Az alfunkció a lokális gép nevét kérdezi le. Meghívásakor a DS:DX-nek egy 16 byte-os pufferre kell mutatnia, ahová a DOS majd a gép nevét helyezi. CX a visszatéréskor a gép azonosítási számát tartalmazza. Hiba esetén (ha nincs bekapcsolva a hálózat) a Carry magas lesz (= 1) és AX-be 1-et ír a DOS.

02-es alfunkció - nyomtató inicializálása

Az alfunkció beállítja a hálózaton installált nyomtatóra - minden file előtt - elküldendő karaktersorozatot. A meghíváskor BX a hozzárendelési számot (lásd később!), CX a karaktersorozat hosszát, DS:SI pedig az első byte címét tartalmazza.

Ha hiba történt (a hálózat nem üzemel), akkor a Carry flag magas lesz (= 1) és AX-be 1 kerül.

17.3.89 5F funkció - hálózati hozzárendelések beállítása/lekérdezése

Az 5F funkció segítségével a hálózati átirányításokat lehet lekérdezni vagy beállítani. Az AL regiszterben lehet megadni az alfunkció számát, ami 02H, 03H vagy 04H lehet.

A funkció csak a DOS 3.10-es változatától használható!

02H alfunkció - hozzárendelés lekérdezése

Az alfunkció meghívásakor BX a hozzárendelési indexet kell, hogy tartalmazza (0 a lista első eleme), míg DS:SI és ES:DI egy-egy puffer kezdőcíme, ahová majd az alfunkció a lokális, illetve a távoli gép nevét helyezi el. Lehet, hogy a lokális név üres, s egyedül a terminátor 0 byte-ból áll.

Visszatéréskor az alfunkció a nevek beírásán túl, CX-ben adja vissza az 5F03-mal beállított felhasználói kódot. BL értéke 3, ha az átirányított eszköz nyomtató; 4, ha lemezegység.

A hibát a magas (= 1) Carry flag jelzi. Ekkor AX a hiba kódját tartalmazza:

1 = Érvénytelen alfunkció kód
18 = Nincs több file

03H alfunkció - hozzárendelés beállítása

Az alfunkció segítségével lehetőség van nyomtató vagy lemezegység hálózatra történő átirányítására. BL-nek a meghíváskor 3-at kell tartalmaznia, ha nyomtatóról, 4-et, ha lemezegységről van szó. CX-ben adhatjuk meg a felhasználói kódot. DS:SI egy 0 byte-tal lezárt sztringet tartalmaz, ami vagy a nyomtató neve, vagy egy lemezegység betűazonosítója kettősponttal együtt, vagy egy üres string. ES:DI a hálózaton implementált egység alltartalomjegyzék nevét tartalmazza. Ennek alakja a következő:

\gép-név[\út] 00H jelszó 00H

Például a következő Turbo Pascal kifejezés egy alszótárt ad meg:

"\kozpont\main\program" + chr(0) + "mancie" + chr(0)

A fenti megadásban a távoli gép neve kozpont, azon a main\program nevű altartalomjegyzékbe irányítjuk át a lemezegység műveleteit.

Ha BL = 3, akkor a DS:SI által megadott név csak PRN, LPT1, LPT2 vagy LPT3 lehet. Ha BL = 4, akkor csak egy érvényes lemezegységet adhat meg, vagy üres lehet. Ez utóbbi esetben az adott jelszóval a hálózati tartalomjegyzéket kísérli meg elérni a rendszer.

Visszatéréskor a Carry magasra (= 1) állítása jelzi, hogy hiba történt, ekkor AX tartalmazza a hiba kódját:

1 = a hálózat nem üzemel, vagy a paraméterek nem megfelelőek
3 = a megadott elérési út nem megfelelő
5 = a hálózati tartalomjegyzék/jelszó kombináció nem megfelelő
8 = nincs elég memória a sztringműveletek elvégzésére.

04H alfunkció - hozzárendelés törlése

A funkció segítségével törölhetünk egy, az 5F03 alfunkció segítségével létrehozott hozzárendelést. DS:SI tartalmazza az átirányított egység nevét. Ez vagy a PRN, LST1, LST2, LST3 nevek egyike, vagy egy érvényes lemezegység betűazonosítója a kettősponttal. A sztring végét egy 0 byte jelzi. Ha a string mindössze két \\ jelből áll, akkor ennek hatására megszűnik a távoli géppel való kapcsolat.

Hiba esetén a Carry magas (= 1) lesz, s AX tartalmazza a hiba kódját:

1 = A hálózat nem üzemel, vagy nem átirányított egységre hivatkozunk;
15 = A nyomtató vagy lemezegység átirányítása a távoli gépen szünetel.

17.3.90 62 funkció - program-szegmens lekérdezése

A funkció visszatéréskor BX-ben a hívó processzor program-szegmens terület szegmens címét adja vissza. Hibajelzés nincs.

Példa:

```
function GetPSP: integer;
begin
   reg.AH:=$62;
   MSDos(reg);
   GetPSP:=regBX
end;
```

17.3.91 65 funkció - kibővített ország-specifikus információk lekérdezése

A DOS 3.30-as verziójától kezdve lényegesen bővült az ország-specifikus információk köre, pl. a kódlapok használatával. A bővebb adatok lekérdezésére egy új DOS funkciót kellett beilleszteni a rendszerbe.

A funkció meghívása előtt BX-be a kódlap sorszámát, DX-be pedig az ország kódját kell elhelyezni. Ha BX = -1, az a használt CON eszköznek felel meg. Hasonlóan, ha DX = -1, akkor a rendszer alapértelmezésében szereplő ország-kóddal dolgozik a funkció. ES:DI a memóriában levő pufferre mutat, CX a puffer hosszát adja. Ennek legalább 5-nek kell lennie. AL-be annak megfelelően az 1,2,4 vagy 6 értékek egyikét kell beírnunk attól függően, hogy milyen adatot kérünk. Az egyes értékek jelentése a következő:

AL	**Jelentés**
1	ország-specifikus információ
2	nagybetű konvenció
4	nagybetű konvenció file-nevekben
6	rendezési tábla

AL értékétől függően a pufferben más és más információt kapunk vissza.

Ország-specifikus információ (AL = 1)

1 byte	01
2 byte	a puffer mérete
2 byte	ország-kód
2 byte	kódlap sorszáma
2 byte	dátum formátuma
5 byte	pénznem jele
2 byte	ezresek elválasztó jele
2 byte	tizedes elválasztó jel
2 byte	dátum elválasztó jel
2 byte	idő elválasztó jel
1 byte	jelző bitek
1 byte	pénzegység használatakor a tizedes jegyek száma
1 byte	idő formátuma
2 duplaszó	a kisbetűk/nagybetűk átalakító rutin kezdőcíme
2 byte	adatelválasztó jel
10 byte	fenntartott (a DOS használja)

Nagybetű konverzió (AL = 2)

1 byte	02
1 duplaszó	a nagybetű tábla elejére

Nagybetű konverzió file-nevekre (AL = 4)

1 byte	04
1 duplaszó	a nagybetű tábla elejére

Mindkét esetben a tábla 130 byte-ból áll. Az első két byte a táblázat méretét adja meg. A táblázat n-ik bejegyzése az n + 128 ASCII karakterhez tartozó nagybetűt adja meg.

Rendezési tábla (AL = 6)

1 byte	06
1 duplaszó	a rendezési tábla elejére

A rendezési tábla 258 byte-ból áll. Az első két byte a tábla hosszát tartalmazza, majd 256 ASCII karakter következik a megfelelő (tehát ABC és nem kód-szerinti) sorrendben.

17.3.92 66 funkció - globális kódlap beállítása/lekérdezése

A funkció segítségével a rendszer által használt kódlapot állíthatjuk be, vagy kérdezhetjük le. Lekérdezéskor AL = 1. Visszatéréskor a Carry-flag magas, ha hiba történt. BX tartalmazza az aktuális kódlapot, míg DX a rendszer indításakor használt kódlapot.

Beállításhoz az AL = 2 értéket kell választani. A rendszer az új kódlapot a COUNTRY.SYS file-ból tölti be a memóriába. Így a kódot külön nem kell megadni. Ha hiba történt, akkor visszatéréskor a Carry-flag magas lesz. *Megjegyezzük, hogy a funkció használata előtt az NLSFUNC program rezidens részét a memóriába kell tölteni.*

Csak a DOS 3.30-as változatától használható!

17.3.93 67 funkció - periféria-kezelők maximális számának megadása

A funkció meghívása előtt a használni kívánt periféria-kezelők számát a BX regiszterbe kell tölteni. Hiba esetén a Carry flag magas lesz. Ezzel a funkcióval maximum 64K-s periféria-kezelőt használhatunk. Ha 20-nál kisebb értéket írunk a BX regiszterbe, akkor azt 20-ra módosítja a rendszer. Megjegyezzük, hogy a CONFIG.SYS file-ban elhelyezett FILES = konfigurációs parancs segítségével maximum 255 periféria-kezelőt tudunk engedélyezni. A DOS részére helyet kell biztosítani, ahol tárolja a bővebb listát. Ezt a 4A funkcióval tehetjük meg.

Csak a DOS 3.30-as változatától használható!

17.3.94 68 funkció - puffer kiürítése

A funkció segítségével adott file-hoz tartozó puffereket a lemezre írhatjuk. BX-be kell elhelyezni a periféria-kezelő sorszámát.

Csak a DOS 3.30-as változatától használható!

A. függelék

A DOS üzenetei

Ez a függelék két részre tagolódik. Az első részben a perifériákkal kapcsolatos hibaüzenetek találhatók. Azok az üzenetek, amelyekkel a DOS felhasználórendszeréhez tartozó perifériás egységeken (például: nyomtató, képernyő, lemezmeghajtó stb.) történő írás vagy olvasás folyamán fellépő hibákat jelzi.

A második részben a DOS egyéb üzenetei találhatók. Mivel a német és angol nyelvű rendszereket használják hazánkban, ezért a hibaüzeneteket mindkét formában megadjuk. Az angol és német nyelvű üzenetek után - gondolatjelek közt - megadjuk az üzenet fordítását. Ez a legtöbb esetben tartalmi, nem szó szerinti fordítás. Ezután közöljük a hiba okát és a kijavítás módját is megadjuk a magyarázat részben. Azt is közöljük, hogy melyik program, parancs küldte az üzenetet.

A DOS egyéb hibaüzenetei a DOS 3.20-as verzió hibaüzenetei. Ennél korábbi verziókban bizonyos üzenetek még nem léteztek. A hibaüzenetek egy része bizonyos korlátozásokhoz kapcsolódik (pl.a szegmensek száma nem haladhat meg egy bizonyos értéket). A különböző verziók esetén ez az érték más és más lehet.

A.1 Perifériákkal kapcsolatos hibaüzenetek

Ha a felhasználó rendszerében perifériaműveletek (írás, olvasás) során a DOS hibát érzékel, akkor a következő formában küld üzenetet:

<TYPE> ERROR READING <DEVICE>
ABORT, RETRY, IGNORE?
(ABBRECHEN, WIEDERHOLEN, IGNORIEREN?)

vagy

<TYPE> ERROR WRITING <DEVICE>
ABORT, RETRY, IGNORE?
(ABBRECHEN, WIEDERHOLEN, IGNORIEREN?)

- <TÍPUS> HIBA AZ OLVASÁS SORÁN <EGYSÉG>
- BEFEJEZNI, ISMÉTELNI, ÁTLÉPNI?

vagy

- <TÍPUS> HIBA AZ ÍRÁS SORÁN <EGYSÉG>
- BEFEJEZNI, ISMÉTELNI, ÁTLÉPNI?

Az ilyen típusú üzenetekben az <EGYSÉG> a hibás egység nevét jelöli, mint például PRN vagy B:. A <TÍPUS> a következő hibatípusokat jelölheti:

BAD CALL FORMAT (FALSCHER AUFRUFFORMAT)
- hiba a hívás formájában

A perifériakezelőnek egy nem megfelelő hosszúságú parancs lett átadva. A hibát ki kell javítani, s a rendszert és a programot újraindítani.

BAD COMMAND (FALSHER BEFEHL)
- hibás parancs

A perifériakezelő az <EGYSÉG>-re hibás parancsot adott ki. A programot kijavítás után lehet csak újrafuttatni.

BAD UNIT (FALSHE EINMEIT)
- hibás meghajtó

A perifériakezelő egy nem létező al-egységre (meghajtóra) hivatkozott. A programot kijavítás után lehet csak újraindítani.

DATA (DATAN)
- adathiba

Az írás vagy olvasás során adathiba fordult elő. Ez általában azt jelenti, hogy fizikailag hibás rész van a lemezen.

DRIVE NOT READY (NICHT BEREIT)
- a lemezegység nincs készenléti állapotban

A megadott egységben:
- nincs lemez;
- nincs bekapcsolva;
- nincs becsukva az ajtó;
- nem megfelelően formázott lemez van.

FCB UNAVAILABLE
- FCB nem használható

A file megosztási környezet betöltése után, az FCBS konfigurációs parancsban megadottnál több file-t akarunk megnyitni.

GENERAL FAILURE
- ismeretlen hiba

A rendszer nem tudta közelebbről azonosítani a hibát. A legvalószínűbb, hogy inkompatibilis, vagy nem megfelelően formázott lemezt használtunk. Próbálkozzunk először az ismétléssel (RETRY, WIEDERHOLENE)

NO PAPER (PAPIERENDE)
- a papír elfogyott

A nyomtatóból kifogyott a papír, nincs bekapcsolva, vagy OFF-LINE üzemmódban van.

NON-DOS DISK (KEINE DOS DISKETTE)
- nem DOS lemez

A file-leíró táblázat (FAT) hibát tartalmaz. Újra kell a lemezt formázni.

NOT READY (NICHT BEREIT)
- nem kész

Ha az üzenet a nyomtatóra vonatkozik, akkor az még nyomtat. Ha a lemezegységre, akkor annak ajtaja nyitva van, vagy nem formázott lemez van benne.

READ FAULT (LESEFEHLER)
- olvasási hiba

A DOS nem tudta sikeresen elolvasni az adatokat.

SECTOR NOT FOUND (SECTORE NICHT GEFUNDEN)
- a szektor nem található meg

Az adatokat tartalmazó szektort nem lehetett a lemezen lokalizálni. Ez akkor történhet, ha a lemezen hibás területek vannak.

SEEK (SUCHEN)
- keresési hiba

A fix vagy hajlékony lemez nem tudja a megadott sávot megtalálni.

SHARING VIOLATON
- megosztási hiba

SHARE. A file-t olyankor kíséreltük megnyitni, amikor az zárolt volt. Először kísérletezzünk az ismétléssel.

WRITE FAULT (SCHEIBFEHLER)
- íráshiba

Az adatokat nem tudta a DOS az adathordozóra írni.

WRITE PROTECT (SCHREIBSCHUTZ)
- írásvédelmi hiba

Írásvédett lemezre akartunk írni.

Megjegyzés:

Ha az üzenetek közül valamelyik a lemezmeghajtóra vonatkozik, akkor a lemezt nem szabad lecserélni. A rendszer addig várakozási állapotba kerül, amíg a felhasználó a következő válaszok valamelyikét meg nem adja:

A/A - befejezés. A rendszer befejezi azt a programot, amely a műveletet végezte.

R/W - ismétlés. A rendszer újra megismétli a műveletet. Érdemes először ezzel próbálkozni.

I/I - átlépés. A hibát átlépi, a program végrehajtását folytatja. Ezt nem javasoljuk, mivel adatok veszhetnek el.

A.2 A DOS egyéb üzenetei

A...

ABORT EDIT (Y/N)?
- Szerkesztés megszakítása (I/N)?

EDLIN. A Q(uit) parancs kiadása után visszakérdez a rendszer, hogy biztos-e, hogy abbahagyjuk a szerkesztést. Y megnyomására visszatérünk a DOS-hoz, N-re folytathatjuk a szerkesztést.

ABOUT TO GENERATE .EXE FILE CHANGE DISKS <HIT ENTER>
(BEREIT ZUM GENERIEREN DER .EXE-DATEI. DISKETTE AUSTAUSCHEN <EINGABE BETATIGEN>)
- Kész az .EXE kiterjesztésű file generálására. Cserélje ki a lemezt, <adjon ENTER-t>

LINK. Ez az üzenet akkor jelenik meg, ha a /PAUSE paramétert megadjuk. Ebben az esetben az adatállományt tartalmazó lemezt a meghajtóba kell helyezni, és le kell nyomni az ENTER billentyűt.

ACCESS DENIED
- Az elérés tilos

Az üzenetet akármelyik parancs küldheti. Valamelyik file-t, alszótárat, vagy perifériát nem megengedett módon próbáltunk használni, például csak olvasható file-ba akartunk írni.

ACTIVE CODE PAGE
- Aktív kódlap

CHCP. A DOS által használt kódlap sorszáma. Egyes periféria-kezelők ettől eltérő kódlapot használhatnak. Infofrmációs üzenet.

ACTIVE CODE PAGE FOR DEVICE DDD IS NNN
- Aktív kódlap a ddd egységen:nnn

MODE. A MODE parancs segítségével valamelyik periféria-kezelő állapotát kérdeztük le. Információs üzenet.

ACTIVE CODE PAGE NOT AVAILABLE FROM CDN DEVICE
- Nincs aktív kódlap a konzolon

KEYB. A KEYB segítségével aa CON: ,konzol állapotát kérdeztük le. A program nem tudta a kódlap sorszámát megállapítani. Ennek oka, hogy a kódlapokhoz szükséges meghajtó nincs installálva, vagy, hogy nem töltöttünk be kódlapot.

ADD d:\út\file-név?
- Hozzáadja a d:\út\file-név file-t?

REPLACE. A /P paraméter megadásakor a parancs visszakérdez, hogy hozzáadja-e a file-t, vagy sem. Az Y az igen, az N a nem válasz.

ADDING d:\út\file-név
- d\út\file-név hozzáadva

REPLACE. A szóbanforgó file-t a rendszer az alszótárhoz adta

ALL FILES CANCELED BY OPERATOR
(ALLE DATEINEN VON BEDIENER GESTRICHEN)
- Valamennyi specifikált file nyomtatását törölte az operátor

PRINT. Ez az üzenet akkor jelenik meg, ha a felhasználó a file-ok kinyomtatását a /T paraméterrel megszakítja.

ALL LOGICAL DRIVES DELETED IN THE EXTENDED DOS PARTITION
- Valamennyi logikai egységet töröltük a kiterjesztett DOS partícióból

FDISK. A kiterjesztett DOS partíciónnincs egyetlen logikai egység sem. A partíciót célszerű törölni. Információs üzenet.

ALL SPECIFIED FILE(S) ARE CONTIGUOUS
(ALLE ANGEGEBENEN DATEINEN SIND ZUSAMMENHÄNGEND)
- Valamennyi specifikált file folyamatosan helyezkedik el

CHKDSK. A megadott file-ok egybefüggő területen vannak a lemezen elhelyezve.

ALLOCATION ERROR FOR FILE, SIZE ADJUSTED
(ZUORDNUNGSFEHLER, GRÖßE BERICHTIGT)
- File lefoglalási hiba, méret javítva

CHKDSK. Az üzenet előtt egy azonosító áll. A file-leíró táblázatban (FAT) érvénytelen szektorszám található. A file-t csonkítja (rövidíti).

AMBIGUOUS SWITCH:Z
(ZWEIDEUTIGER SCHALTER:Z)
- Kettős jelentésű kapcsoló

LINK. A Z-vel megadott jelek nem egyértelműen LINK paraméterek. Ebben az esetben a paraméter azonosítóból több jelet kell megadni.

AMOUNT READ LESS THAN SIZE IN HEADER
- A valódi méret kisebb a fejlécben megadottnál

EXE2BIN. A program file hossza kevesebb volt, mint amit a fejléc jelzett. Feltehetően újra kell szerkeszteni a programot.

APPEND ALREADY INSTALLED
- Az APPEND rezidens részét már installáltuk

APPEND. Az APPEND-et csak egyszer lehet a tárba tölteni, mi másodszor is megkíséreltük. Az APPEND-et használhatjuk.

APPEND/ASSIGN CONFLICT
- APPEND és ASSIGN helytelen együttes használata

APPEND. Az ASSIGN parancsot az APPEND előtt használtuk. Csak fordított sorrendben lehet.

APPEND/TOPVIEW CONFLICT
- APPEND és TOPVIEW helytelen együttes használata

APPEND. A TOPVIEW és az APPEND programokat rossz sorrendben indítottuk. Ki kell lépni a TOPVIEW-ból, betölteni az APPEND-et, és újból indítani a TOPVIEW-t.

AN INTERNAL FAILURE HAS OCCURED
(EIN INTERNER FEHLER IST AUFGETRETEN)
- Belső hiba a szerkesztés folyamán

LINK. A LINK működése alatt hiba jelentkezett. Ha ismétlődik, akkor a LINK programot le kell cserélni. A bibát a rendszer forgalmazójának is jelenteni kell.

ARRAY ELEMENT SIZE MISMATCH
- A tömb mérete eltér

LINK. Egy távoli (FAR) közös tömböt eltérő mérettel deklaráltunk.

ATTEMPT TO ACCESS DATA OUTSIDE OF SEGMENT SOUNDS
- Szegmenshatáron túl akartunk lépni

LINK. Hibás object file.

ATTEMPT TO PUT SEGMENT XXXXXX IN MORE THAN ONE GROUP IN FILE XXXXXX
- Az xxxxxx szegmens az xxxxxx file-ban két csoportban van.

LINK. Egy szegmenst két különböző csoportban deklaráltuk.

ATTEPMPTED WRITE - PROTECT VIOLATION
(SCHREIBSCHUTZ - SCHREIBEN NICHT MÖGLICH)
- Kísérlet történt az írásvédelem megsértésére

FORMAT. Az éppen formázás alatt álló lemez írásvédelemmel rendelkezik. A lemezt ki kell cserélni, majd akármelyik billentyű lenyomásával folytatható a formázás.

B...

**** BACKING UP FILES TO TARGET DRIVE x****
TARGET NUMBER: y
***** BACKING UP FILES TO DRIVE x *****
NUMBER: y
- Az x egységről az y file mentése megkezdődött

BACKUP. Az elmentett file sorszámát jelző üzenet. Hiba nem történt.

BACKUP FILE SEQUENCE ERROR
(DISKETTE NICHT FOLGERICHTING)
- A lemez nem megfelelő

RESTORE. Egy ismételten előállítandó file több lemezen található. A felhasználó nem azt a lemezt helyezte be, amelyen az állomány első része található. A helyes lemez betétele után a RESTORE-t újra kell indítani.

BAD COMMAND OR FILE NAME
(FALSHER BEFEHL ODER DATEINAME)
- Helytelen parancs vagy file-név

DOS. Az éppen beadott parancs érvénytelen. Adjuk ki újra, helyesen! Ha a parancs korrekt, akkor ellenőrizni kell, hogy a parancshoz tartozó file(-ok) a megadott lemezen van(nak)-e

BAD INTERNAL RELOC TABLE
- Rossz belső relokációs tábla

LINK. A programszerkesztő belső hibája.

BAD NUMERIC PARAMETER
(FALSHER NUMERISCH PARAMETER)
- Hibás numerikus paraméter

LINK. A /STACK paraméterben megadott érték nem érvényes numerikus érték.

BAD OR MISSING COMMAND INTERPRETER
(FALSHER ODER FEHLENDER BEFEHLSINTERPRETER)
- Helytelen vagy hiányos parancsértelmező

DOS. Az a lemez, amelyről a DOS indult, nem tartalmazza a COMMAND.COM-ot, vagy a rendszer betöltésénél hiba történt. Ha a problémát a rendszer visszatöltésével nem lehet megoldani, akkor a COMMAND.COM-ot a háttér lemezről ismételten át kell másolni.

Ez az üzenet jelenik meg akkor is, ha a tartalomjegyzékből töröltük a COMMAND.COM bejegyzést. Ezt az üzenetet kapjuk akkor is, ha a COMSPEC = xx paraméter tartalomjegyzékhez vezető olyan környezetet ad meg, amely a COMMAND.COM adatállományt nem tartalmazza, és a DOS megkísérli a parancsprocesszor ismételt behívását.

BAD OR MISSING <file-név>
(FALSHER ODER FEHLENDER (DATEINAME))
- Helytelen vagy hiányzó file-név

DOS. Ez az üzenet csak a rendszer indításánál jelenik meg. Azt jelenti, hogy egy periféria-kezelő program, amelyet a COMFIG.SYS file-ban specifikáltunk, nem található. Előfordulhat még, hogy a megadott megszakítási cím (interrupt adress) a meglévő konfiguráció címtartományán kívül esik. Ezt a perifériakezelő programot a DOS nem installálja.

BAD OR MISSING KEYBOARD DEFINITION FILE
- Rossz vagy hiányzó billentyűzet definíciós tábla

KEYB. A program nem találta a KEYBOARD.SYS file-t, vagy az rossz adatokat tartalmazott.

BAD PARTITON TABLE
- Hibás partíciós tábla

FORMAT. A lemez partíciós táblája hibás, vagy nem tartalmaz DOS partíciót.

BAD UNIT
- Hibás meghajtó

Bármelyik parancs küldheti. Valamelyik lemezes perifériakezelő egy kritikus hibát talált valamelyik egységen.

BAD TRACKS FOUND AT START OF PARTITION
PARTITION SIZE ADJUSTED
- Hibás sávok miatt a partíció mérete módosult

FDISK. A fix lemez DOS által igényelt részén hibás blokkok vannak. A rendszer ezeket átlépte, ezzel a partíció mérete csökkent.

BATCH FILE MISSING
(STAPELVERARBEITUNGSDATEI FEHLT)
- A kötegelt (batch) file hiányzik

DOS. A DOS a parancs file-t nem találta az illető .BAT file feldolgozása közben. Ez valószínűleg törlés vagy átnevezés miatt történt. A feldolgozás befejeződik és a DOS rendszerazonosító (prompt) jelenik meg.

BF

DEBUG. Hibás azonosítójel. A kódot hibásan adtuk meg. A regiszter parancsot (RF) a helyes értékkel ismételten ki kell adni.

BP

DEBUG. Teszt leállások (break-points). Több, mint tíz teszt leállást adtunk meg a GO parancs számára. A GO parancsnak legfeljebb tíz teszt leállást adjuk meg.

BR

DEBUG. Hibás regiszter azonosítót adtunk meg. Az R parancsot helyes azonosítóval ismételten ki kell adni.

BREAK IS ON/OFF
- CTRL-BREAK használata BE/KI kapcsolva

BREAK. Az üzenet jelzi, hogy a <CTRL> + <Break> leütésének van-e azonnali hatása vagy sem. Nem hiba.

BUFFER SIZE ADJOUSTED
- Puffer mérete módosítva.

VDISK. A parancs módosította a CONFIG.SYS-ben megadott bufferméretet.

BUFFER SIZE	Puffer méret
SECTOR SIZE	Szektor méret
DIRECTORY ENTRIES	Tartalomjegyzék nagysága
TRANSFER SIZE	Egyszerre átvihető sávok száma

C...

CANNOT CHDIR TO ROOT
- A főtartalomjegyzék sérült

CHKDSK. A CHKDSK parancs végrehajtása félbeszakadt, mert a DOS nem tudott egy CD\ parancsot végrehajtani. A lemez sérült.

CANNOT CHKDSK A NETWORK DEVICE
CANNOT CHKDSK A NETWORK DRIVE
- A parancs hálózaton implementált lemezegységre nem hajtható végre.

CHKDSK.

CANNOT CHKDSK A SUBSTED OR ASSIGNED DRIVE
- SUBST és ASSIGN után a CHKDSK nem használható

CHKDSK. A SUBST parancs elérhetetlenné teszi a CHKDSK számára szükséges információt, ezért az nem hajtható végre.

CANNOT CREATE EXTENDED DOS PARTITION WITHOUT PRIMARY DOS PARTITION ON DISK 1

- Az első fix lemezen nem hozható létre kibővített DOS partíció az elsődleges DOS partíció nélkül.

FDISK. Az 1-es fix lemezen kibővített DOS partíciót akartunk létrehzni, de még nem hoztunk létre elsődlegesDOS partíciót. Először ez utóbbit kreáljuk meg, s csak utána a kibővítettet.

CANNOT CREATE A ZERO CILINDER PARTITION

- 0 cilinderből álló partíció nem hozható létre

FDISK. 0 cilinderből álló partíciót próbáltunk meg létrehozni (valószínűleg tévedésből). Adjuk meg a helyes partícióméretet, és ismételjük meg a funkciót.

CANNOT DELETED EXTENDED DOS PARTITION WHILE LOGICAL DRIVES EXIST

- Logikai egységet tartalmazó kibővített DOS partíciót nem törölhetünk.

FDISK. A kibővített DOS partíció törlése előtt az abban levő valamennyi logikai egységet törölni kell. Ha ezt elfelejtettük, akkor kapjuk ezt az üzenetet.

CANNOT DISKCOMP TO OR FROM A NETWORK DRIVE

- DISKCOMP hálózaton nem használható

DISKCOMP. A parancs nem használható hálózaton implementált lemezegységre még akkor sem, ha az fizikailag ugyanazon a számítógépen van. Ehelyett a COMP *.* parancsot célszerű használni.

CANNOT DISKCOPY TO OR FROM A NETWORK DRIVE

- DISKCOPY hálózaton nem használható

DISKCOPY. A DISKCOPY parancs nem használható hálózaton implementált lemezegységre még akkor sem, ha az fizikailag ugyanazon a számítógépen van. Ehelyett a COPY *.* parancsot lehet használni.

CANNOT DO BINARY READS FROM A DEVICE
(BINARES LESEN VON EINER EINHEIT UNMÖGLICH)

- A bináris olvasás nem hajtható végre az egységről

COPY. A /B paramétert egységnévvel használtuk (Pl. CON:) a másoláshoz. A másolás bináris módban nem hajtható végre, mert a COPY-nak (EOF) jelet kell kapnia az egységről. Ekkor a /B paramétert el kell hagyni, vagy az egységnév után a /A paramétert kell megadni.

CANNOT EDIT .BAK FILE - RENAME FILE
(.BAK-DATEI KANN NICHT EDITIERT WERDEN - DATEI UMBENNEN)

- A .BAK kiterjesztésű file-ok védett állományok, ezért ezek nem szerkeszthetők. Ezeket csak átnevezés után lehet szerkeszteni.

CANNOT EXEC BASICA.COM
- A BASICA.COM file nem tölthető be

BASIC. A ROM BASIC nem tudja elindítani a BASICA.COM file-t. Feltehető, hogy a file nincs az elérhető szótárak egyikében sem.

CANNOT EXECUTE FORMAT
- FORMAT nem hajtható végre

SELECT. A FORMAT végrehajtása közben hiba történt. Elképzelhető, hogy a parancsot nem találja a rendszer.

CANNOT EXECUTE XCOPY
- XCOPY nem hajtható végre

SELECT. Az XCOM végrehajtása során hiba történt. Elképzelhető, hogy a parancsot nem találja a rendszer.

CANNOT FDISK A NETWORK DRIVE
- FDISK hálózaton nem használható

FDISK. A hálózaton implementált fix lemezegységet nem lehet átparticionálni.

CANNOT FIND FILE <OBJEKT FILE> CHANGE DISKETTE AND PRESS <ENTER>
(DATEI NICHT GEFUNDEN - OBJEKTDATEI DISKETTE AUSTAUSCHEN <EINGABE BETTIGEN>)
- A tárgy modul nem található. Cserélje ki a lemezt

LINK. A LINK program a megadott tárgy (objekt) modult nem találta. A megadott modult tartalmazó floppy-t kell betenni, majd az <ENTER> billentyű lenyomásával a LINK folytatja a programszerkesztést.

CANNOT FIND LIBRARY FILE ENTER NEW DRIVE LETTER
(BIBLIOTHEK NICHT GEFUNDEN - BIBLIOTHEKSDATEI NEUE LAUFWERK-BUCHSTABEN EINGEBEN)
- A könyvtári file nem található. Adja meg az új meghajtó azonosítót.

LINK. A megadott könyvtár (.LIB) nem található a lemezen. Azt az új meghajtó azonosítót kell beadni, amelyen a könyvtár található.

CANNOT FIND SYSTEM FILES
- A rendszer file-ok nem találhatók

FORMAT vagy SYS. Az IBMBIO.COM és/vagy IBMDOS.COM file-okat nem találja a program.

CANNOT FORMAT A NETWORK DRIVE
- FORMAT hálózaton nem használható

FORMAT. Hálózaton implementált lemezegységet nem lehet megformázni.

CANNOT FORMAT A SUBSTED OR ASSIGNED DRIVE
- SUBST és ASSIGN után a FORMAT nem használható

FORMAT. A FORMAT parancs kiadása előtt nem használhatjuk a SUBST és ASSIGN parancsokat. Az ASSIGN segítségével állítsuk vissza az eredeti állapotot és utána formázzuk meg a lemezt.

CANNOT FORMAT NONREMOVABLE DRIVE X
- Az x fix egység nem formázható

BACKUP. A parancs használatakor az /F opciót is megadtuk, holott az fix egységre nem használható. Ha mégis szükség van a fix lemez megformázására, azt a BACKUP parancs előtt, a FORMAT-tal végezzük el.

CANNOT JOIN TO A NETWORK DRIVE
- JOIN hálózaton nem használható

JOIN. A JOIN segítségével nem kapcsolhatunk össze lokális és hálózaton implementált lemezegységet.

CANNOT LABEL A NETWORK DRIVE
- LABEL hálózaton keresztül nem használható

LABEL. A parancsot nem adhatjuk ki egy hálózaton implementált lemezegységre.

CANNOT LOAD COMMAND, SYSTEM HALTED
(COMMAND KANN NICHT GELADEN WERDEN, SYSTEM ANGEHALTEN)
- A COMMAND .COM nem tölthető, rendszer leállítva

DOS. A parancsprocesszor ismételt töltése során a DOS megállapította, hogy a lemezterület megsérült. Ezt az üzenetet kapjuk akkor is, ha a parancsprocesszor a COMSPEC=xx környezet-paraméterrel megadott úton nem található. Ebben az esetben a rendszerlemezről ismét töltsük be a DOS-t.

CANNOT NEST RESPONSE FILE
(ANTWORTDATEI KANN NICHT VERSCHACHTELT WERDEN)
- Az automatikus válasz file-t nem tudta besorolni

LINK. A < @FILE-SPEC > -t, az automatikus válasz-file-t használtuk.

CANNOT OPEN LIST FILE
(LISTDATEI KANN NICHT ERÖFFNET WERDEN)
- A lista file-t nem tudta megnyitni

LINK. A tartalomjegyzék vagy a lemez megtelt.

CANNOT OPEN OVERLAY
(ÜBERLAGERUNG KANN NICHT ERÖFFNET WERDEN)
- Az OVERLAY nem végrehajtható.

LINK. A tartalomjegyzék vagy a lemez megtelt.

CANNOT OPEN RESPONSE FILE: <file-név>
(ANTWORTDATEI KANN NICHT ERÖFFNET WERDEN)
- A válasz-file-t nem tudja megnyitni.

LINK. Az automatikus válasz-file-t nem találta

CANNOT OPEN RUN FILE

LINK. A tartalomjegyzék vagy a lemez megtelt.

CANNOT OPEN TEMPORARY FILE
(TEMPORAERE DATEI KANN NICHT ERÖFFNET WERDEN)
- Az ideiglenes file-t nem tudta megnyitni.

LINK. A tartalomjegyzék vagy a lemez megtelt.

CANNOT PERFORM A CYCLIC COPY
- Ciklus másolás nem hajtható végre.

XCOPY. A paraméterek alapján a forrás és a cél részben azonos lenne. Pl. a cél egy al-szótár, míg a forrás egy azt tartalmazó másik altartalomjegyzék.

CANNOT RECOVER ENTRY, PROCESSING CONTINUED
- Nem tud belépni a RECOVER-be, az eljárás folytatódik.

CHKDSK. Nem hibaüzenet.

CANNOT RECOVER TO A NETWORK DRIVE

RECOVER. A RECOVER parancs hálózatban implementált lemezegységre nem használható.

CANNOT REOPEN LIST FILE
- A lista file-t nem lehet újra megnyitni

LINK. Az eredeti lemezt nem helyeztük vissza. Újra kell indítani a programszerkesztőt.

CANNOT START COMMAND, EXITING
(COMMAND KANN NICHT GESTARTET WERDEN, BEENDET)
- COMMAND nem indítható, kilépés.

DOS. A parancsprocesszor (COMMAND.COM) második példányának betöltése során megállapította, hogy a FILES = paraméter a konfigurációs adatállományban túl kicsi értéket vesz fel, vagy túl kicsi a tárméret ahhoz, hogy a COMMAND.COM másolatát abban elhelyezzük.

CANNOT USE FASTOPEN FOR DRIVE x
- Az x egységre nem használhatjuk a FASTOPEN parancsot.

FASTOPEN. A JOIN, SUBST vagy ASSIGN parancsokban szereplő, ill. hálózaton implementált egységekre nem adhatjuk ki a FASTOPEN parancsot.

CANNOT USE PRINT - USE NET PRINT

PRINT. A hálózatvezérlő gépen nem adható ki a PRINT parancs. Helyette a NET PRINT parancsot kell használni.

CANNOT SUBST TO A NETWORK DRIVE

SUBST. A parancs nem használható hálózaton implementált egységre.

CANNOT SYS TO A NETWORK DRIVE
- SYS hálózaton nem használható.

SYS. A SYS segítségével a rendszer file-ok nem másolhatók hálózatban implementált lemezegységről vagy lemezegységre.

CANNOT XCOPY FROM A RESERVED DEVICE
- XCOPY fentartott egységről nem használható.

XCOPY. A forrás file helyett egy logikai egységnevet adtunk meg, a másolás nem hajtható végre.

CANNOT XCOPY TO A RESERVED DEVICE
- XCOPY fenntartott egységre nem használható.

XCOPY. A cél file helyett egy logikai egységnevet adtunk meg, a másolás nem hajtható végre.

CHDIR... FAILED TRYING ALTERNATE METHOD
- Az Alternatíva módszer sikertelennek bizonyult.

CHKDSK. A DOS-t újból kell indítani.

CODE PAGE NOT PREPARED
- A kódlap nincs betöltve

MODE. A MODE parancsban egy olyan kódlap sorszámára hivatkozhatunk, amelyiket vagynem aktiváltunk, vagy a kódlap formája nem felel meg a választott kijelzési módnak.

CODE PAGE xxx NOT PREPARED FOR ALL DEVICES
- Az xxx sorszámú kódlapot nem definiáltuk valamennyi egységre.

CHCP. A parancs nem tudta a jelzett kódlapot valamennyi egységre aktualizálni. Ennek többféle oka lehet: az egység nem volt felkészítve erre a kódlapra, átvitelihiba történt, az egység nyomtatott, stb. A hiba elhárítása után a parancs megismételhető.

CODE PAGE xxx NOT PREPARED FOR SYSTEM
- az xxx sorszámú kód-lapot a rendszer nem tudja használni

CHCP. A jelzett sorszámú kód-lapot a parancs nem tudja kiválasztani. Ennek több oka lehet: nem töltöttük be a NLSFUNC programot, a kód-lap az adott ország-specifikus információ mellett nem használható, vagy valamelyik eszközt nem készítettük elő a MODE paranccsal.

CODE PAGE OPERATION NOT SUPPORTED ON THIS DEVICE
- Ezen az egységen a kód-lapok nem használhatók.

MODE. A PREPARE vagy SELECT alparancsot olyan egységre használtuk, amelyik nem képes a kód-lapok kezelésére. Ennek az is lehet az oka, hogy a CONFIG.SYS filében nem megfelelő DEVICE parancsot helyeztünk el.

CODE PAGE REQUESTED yyy IS NOT VALID FOR GIVEN KEYBOARD CODE
- a megadott kód-lap kódja (yyy) az adott billentyűzet-kóddal összeférhetetlen.

KEYB. A KEYB paranccsal olyan kód-lapot kívántunk aktivizálni, amelyik az adott billentyűzet-kóddal nem használható.

CODE PAGE SPECIFIED HAS NOT BEEN DESIGNATED
- a megadott kód-lap nem használható.

KEYB. A parancsban megadott kód-lapot nem definiáltuk. A MODE PREPARE paranccsal ezt tegyük meg, majd ismételjük meg a KEYB-t! Ez az üzenet csak abban az esetben jelenik meg, ha a kód-lapozást a CON:-ra installáltuk.

CODE PAGE SPECIFIED HAS NOT BEEN PREPARED
- a megadott kód-lapot nem készítettük elő.

KEYB. A CON: eszközre installáltuk a kód-lapozást, de a megadott kód-lap használatára nem készítettük elő. A MODE paranccsal ezt végezzük el, majd ismételjük meg a KEYB-t!

CODE PAGE SPECIFIED IS INCONSISTENT WITH INVOKED CODE PAGE
CODE PAGE SPECIFIED IS INCONSISTENT WITH SELECTED CODE PAGE
- a megadott kód-lap összeegyeztethetetlen a meghívott kód-lappal.

KEYB. Figyelmeztetés: a parancsban megadott kód-lap lett az aktuális, de a CON: és a billentyűzet kód-lapja eltér.

CODE PAGE xxx
- kód-lap = xxx

MODE. A MODE segítségével az aktív kód-lapot kérdeztük le. Információs üzenet.

CODE PAGES CAN NOT BE PREPARED
- a kód-lapok nem készíthetők elő.

MODE. A megadott kód-lapok között ugyanaz a sorszám kétszer szerepel, vagy többet adtunk meg a megengedettnél. Ellenőrizzük a kód-lapok sorszámának a beírását, majd ismételjük meg a MODE parancsot.

COMMON AREA LONGER THAN 65536 BYTES
- A közös rész több, mint 65536 byte

LINK. A felhasználói program COMMON változóinak hossza több, mint 64K, ezért a programszerkeztés nem hajtható végre. Jelenleg csak a Microsoft C fordíto okozhatja ezt a hibát.

COMn: bbb,p,d,s,t INITIALIZED
(COMn: bbb,p,d,s,t INITIALISIERT)
- Az aszinkron kommunikációs adaptert a következő értékekkel inicializáltuk

MODE. Az egyes értékek jelentése:
n: adapter azonosító (COM1 vagy COM2)
bbb: átviteli sebesség (band)
p: paritás
  e: páros
  o: páratlan
  n: nincs
s: stopbitek (1 vagy 2)
t: az egység típusa:
  P: nyomtató (időtúllépés esetén újra próbálkozik)
  -:egyéb egység. Nincs ismétlés.

COMPARE ERROR AT OFFSET XXXXXXX
(VERGLEISCHFEHLER BEI OFFSET XXXXXXX)
- Összehasonlítási hiba az OFFSET XXXXXXX-nél.

COMP. Az összehasonlított file-ok a megadott helyen (hexadecimális offset) különböző értékeket tartalmaznak. Az eltérő byte-ok hexadecimális formában kijelzésre kerülnek.

COMMAND FORMAT: DISKCOPY d: d: [/1]
- a parancs alakja DISKCOPY d: d: [/1]

DISKCOPY. Érvénytelen paraméterekkel hívtuk meg, s válaszként kiírja a paraméterek használatának a módját.

COM: ,e,7,1
COMPARE ERROR (3) 04
Track xx, side yy
- Eltérés az yy oldal xx sávján.

DISKCOMP. A jelzett helyen egy vagy több blokkban eltérő adatokat talált a parancs a két lemezen.

COMh: bbbb,p,d,s,t, INITIALIZED
- a COMh csatorna a bbbb,p,d,s,t paraméterekkel üzemkész.

MODE.	Az egyes paraméterek jelentése:
h	adapter (COM1-COMh);
bbbb	átvitel sebessége baud-ban;
p	paritás;
e	páros;
o	páratlan;
n	semmilyen;
d	adatbitek;
s	adatbitek (1,2)
t	a soros eszköz típusa
p	soros nyomtató
-	nincs soros eszköz

COMPARE MORE DISKETTES (Y/N)?
(WEITERE DISKETTEN VERGLEICHEN (J/N)?)
- Kell még lemezeket összehasonlítani (I/N)?

DISKCOMP. Ha további összehasonlítást akarunk, akkor Y-t, ill. J-t kell megadni. Ebben az esetben a DISKCOMP kéri az összehasonlítandó lemezek behelyezését. Ha nem akarjuk a lemezeket összehasonlítani, N-t kell megadni.

COMPARE MORE FILES (Y/N)?
- Kell még file-okat összehasonlítani (I/N)?

COMP. A parancs befejezte a file-ok összehasonlítását és megkérdezi, hogy akarunk-e még továbbiakat összehasonlítani. N-re visszatérhetünk a DOS-ba, az Y újraindítja a COMP parancsot.

COMPARE PROCESS ENDED
- az összehasonlítás befejeződött.

DISKCOMP. Információs üzenet: a DISKCOMP parancs végrehajtása befejeződött.

COMPARING X SECTORS PER TRACK, n SIED(S)
(VERGLICHTEN WERDEN X SEKTOREN PRO SPUR, n SEITE(n))
- Összehasonlításra kerül sávonként x szektor, n oldal

DISKCOMP. Az n értéke 1 vagy 2. Az összehasonlítandó oldalak számát adja. Az x a sávonkénti szektorok száma az első lemezen (8 vagy 9). Ha a /8 paramétert megadtuk, akkor ez az érték 8.

CONFIGURATION TOO LARGE FOR MEMORY
- A konfiguráció túl nagy a memóriához képest.

DOS. A CONFIG.SYS parancsban megadott FILES és BUFFERS allokálása után nem maradt elég memória a DOS betöltésére. Előfordulhat, hogy a COMMAND-ban szerepel a /E paraméter , és az abban megadott méret kevés a COMMAND.COM file betöltésére.

CONTAINS INVALID CLUSTER, FILE TRUNCATED
(DATEI HAT UNGÜLTINGEN BEREICH, DATEI ABGESCHNITTEN)

- A file érvénytelen területet tartalmaz, a file-t megcsonkította.

CHKDSK. A file érvénytelen adatblokkot tartalmaz. A hibás blokkot levágja /F paraméter esetén.

CONTAINS XXX NON-CONTIGUOUS BLOCKS
(DATEI ENHÄLT XXX NICHT ZUSAMMENHÄNGENDE BLÖCKE)

- A file nem összefüggő blokkot tartalmaz.

CHKDSK. A megadott file fizikailag nem összefüggő területen helyezkedik el, hanem xxx darabban. Ez informatív üzenet. A felszeletelt file feldolgozása hosszabb időt vesz igénybe, ezért érdemes esetleg egy másik lemezre átmásolni (COPY). Ekkor a file feltehetően kevesebb darabban kerül a lemezre, és így gyorsabb lesz az elérése.

CONVERT DIRECTORY TO FILE (Y/N)?
(INHALTSVERZEICHNIS IN DATEI UMWANDELN (J/N)?)

- A tartalomjegyzék átalakítása a file-hoz (I/N)?

CHKDSK. A megadott tartalomjegyzék már annyi hibás bejegyzést tartalmaz, hogy az már nem használható directory-ként. Ha Y ill. J választ adunk, akkor a rendszer a file-ok listáját tartalomjegyzékké alakítja, és DEBUG programmal átvizsgálhatóvá válik. Ha N-t adunk meg, akkor a bejegyzések nem változnak a tartalomjegyzékben.

CONVERT LOST CHAINS TO FILES (Y/N)?
(VERLORENE BLÖCKE IN DATEINEN UMVANDELN (J/N)?)

- Az elveszett láncok file-okká alakítása (I/N)?

CHKDSK. Ha a hibás blokkokban található file-okat ismét elő akarjuk állítani, akkor Y-t ill. J-t kell megadni. Ha N-t adunk meg, akkor a CHKDSK törli az adathibás blokkokat.

COPY ANOTHER (Y/N)?
(WEITERE KOPIE ERSTELLEN (J/N)?)

- Szükséges még másik COPY (I/N)?

DISKCOPY. Ha újabb teljes lemezt akarunk másolni, akkor Y-t, ill. J-t adunk meg. A DISKCOPY ezután kéri a szükséges lemez behelyezését. Ellenkező esetben N-t kell megadni.

COPY COMPLETE
(KOPIEREN BEENDET)

- A másolás befejeződött.

DISKCOPY. A másolás a forráslemezről a céllemezre sikeresen befejeződött.

COPYING X SECTORS PER TRACK, N SIDE
(KOPIEREN VON X SEKTOREN PRO SPUR, N SEITE)
- Másolás X szektor sávonként, N oldal

DISKCOPY. Az N 1 vagy 2 lehet. Ez az oldalak számát adja meg, amit a forráslemez első sávjáról olvasott le a DISKCOPY. Az X 8 vagy 9 lehet, a forráslemez sávonkénti szektorainak száma.

CURRENT CODE PAGE SETTINGS
- aktív kód-lapok beállítása.

MODE. A MODE GET GLOBAL parancsra válaszként ezzel a szöveggel kezdődő listát kapunt. Információs üzenet.

CURRENT DRIVE IS NO LONGER VALID
- Az aktuális lemezegység hibás.

COMMAND. Miközben a DOS készenléti jelét ($p) akarta kijelezni, a COMMAND érzékelte, hogy az aktuális lemezegység nem tartalmaz érvényes információt, pl. töröltük a hálózaton implementált egységet.

CURRENT KEYBOARD CODE: xx CODE PAGE: yyy
CURRENT CON CODE PAGE: zzz
- aktív billentyűzet-kód: xx kód-lap: yyy
- aktív CON kód-lap: zzz

KEYB. Információs üzenet a használt billentyűzet-kódról és a használt kód-lapokról.

CURRENT KEYBOARD DOES NOT SUPPORT THIS CODE PAGE
- a kiválasztott billentyűzet nem támogatja a megadott kód-lapot.

MODE. Egy MODE PREPARE végrehajtása során kiderült, hogy az adott feltételek mellett ez a kód-lap nem használható.

D...

DATA RECORD TOO LARGE
- A LEDATA rekord túl nagy.

LINK. A LEDATA rekord több, mint 10224 byte hosszú volt. Fordítási hiba, valószínűleg a fordító helytelen működéséből adódik.

DELETE CURRENT VOLUME LABEL (Y/N)?
- Töröljem a régi címkét (I/N)?

LABEL. Ha nem írjuk be az új címkét, csak az <ENTER>-t nyomjuk meg, akkor kapjuk a fenti üzenetet. Ha nem akarjuk a meglevő címket törölni, akkor az N, különben a Y megnyomásával válaszoljunk.

DEVICE ERROR DURING STATUS
DEVICE ERROR DURING PREPARE
DEVICE ERROR DURING SELECT
DEVICE ERROR DURING WRITE OF FONT FILE TO DEVICE
- hiba az egységen.

MODE. A jelzett művelet közben az egységen hiba történt.

DEVICE OR CODE PAGE MISSING FROM FONT FILE
- az egység vagy a kód-lap hiányzik a font-file-ból.

MODE. A PREPARE alparancs végrehajtása során kiderült, hogy a font-file nem megfelelő. A hiba hatására a kód-lapok definiálatlanná válnak, valamennyit újból elő kell készíteni.

DEVICE ddd NOT PREPARED
- a ddd sorszámú kód-lap nincs előkészítve.

MODE. A MODE paranccsal próbáltuk lekérdezni a ddd egység állapotát. Eszerint az egység egyáltalán nem lett a kód-lapozásra előkészítve.

DF

DEBUG. Dupla jelölés. Ellentétes kódokat használtunk a jelölésre. Regiszter parancs (RF) használatánál egy jelölést csak egyszer lehet felhasználni.

DIR PATH LISTING FOR VOLUME xxxxxxx
- Az xxxxxxx kötet útjai következnek.

•TREE. Az üzenet a lemez címkéjét írja ki.

DIRECTORY ENTRIES ADJUSTED
- Tartalomjegyzék nagysága módosítva

VDISK. A VDISK parancs módosította a DEVICE = VDISK.SYS parancsban megadott tartalomjegyzékek maximális számát.

DIRECTORY IS JOINED, TREE PAST THIS POINTNOT PROCESSED
- JOIN-nal összekötött egységre a CHKDSK nem lép át

CHKDSK. A lemezegységek összekapcsolása a tartalomjegyzékek fa-struktúráját megnöveli. A CHKDSK egy ilyen ponthoz ért, de nem kezdte el a fizikailag másik lemez ellenőrzését. Nem hibaüzenet.

DIRECTORY IS TOTALLY EMPTY, NO . OR .., TREE PAST THIS POINT NOT PROCESSED
- Üres altartalomjegyzék . és .. nélkül, a CHKDSK nem lép bele.

CHKDSK. A parancs olyan altartalomjegyzéket talált, amelyikben sem a ., sem a .. bejegyzés nem szerepel. Valószínűleg az altartalomjegyzék aktivizálása közben rendszerhiba történt. A RECOVER paranccsal meg lehet kísérelni a tönkrement lemezről a file-ok visszaállítását.

DISC BOOT FAILURE
(BOOT FEHLER)
- BOOT hiba

DOS. A DOS memóriába töltése során hiba fordult elő. Ha a rendszer a további próbálkozások során sem indul, akkor egy tartalék rendszerlemezt kell használni!

DISC ERROR READING DRIVE X
- Az x egységen olvasási hiba

A program megpróbált egy fizikai olvasást egy hálózaton implementált lemezegységről. Erre nincs lehetőség.

DISC ERROR WRITING DRIVE X
- Az x egységen íráshiba.

A program megpróbált egy fizikai írást egy hálózaton implementált lemezegységre. Erre nincs lehetőség.

DISC ERROR WRITING FAT X
(PLATTENFEHLER BEI SCHREIBEN DER DATEIZUORDNUNGSTABELLE X)
- Lemez hiba az x FAT írásánál.

CHKDSK. A file-leíró táblázat (FAT) aktualizálása során a CHKDSK lemezhibát talált. Az x 1 vagy 2 lehet aszerint, hogy melyik példányát nem tudta a FAT-ba beírni. Ha ez az üzenet kétszer megjelenik, akkor a lemezt használhatatlannak kell tekinteni. (Lehetséges, hogy a FAT mindkét példánya sérült.)

DISC FULL - WRITE NOT COMPLETED
DISK FULL. EDITS LOST
(DISKETTE VOLL - SCHREIBEN NICHT BEENDET)
- A lemez megtelt - a kiírás nem fejeződött be.

EDLIN. A szerkesztés rendhagyóan fejeződött be, mivel a lemezen nincs elég terület a teljes file kimentésére. A file lemezre mentett része megmarad. de a memóriában levő rész elvész.

DISC NOT COMPATIBLE
(DISKETTE NICHT VERTRÄGLICH)
- A lemez nem kompatibilis

FORMAT. A felhasználó által megadott meghajtón levő lemez a FORMAT paranccsal nem formázható. Ennek oka, hogy a használt lemezegység nem rendelkezik IBM interface-szel.

DISC UNSUITABLE FOR SYSTEM DISC
(DISKETTE NICHT ALS SYSTEMDISKETTE VERWENDBAR)
- A lemez rendszerlemezként nem használható.

FORMAT. A lemez elején hibás sáv található. A lemez DOS rendszer tárolására nem alkalmas, csak adatokéra.

DISKETTE/DRIVE not COMPATIBLE
- A lemez vagy a lemezegység nem kompatibilis.

DISKCOPY vagy DISKCOMP. A cél és forráslemez eltérő formája nem teszi lehetővé a másolás (összehasonlítás) végrehajtását. Pl. az egyik 360 Kbyte-os, míg a másik 1.2 Mbyte-os.

DISKETTES COMPARE OK
(DISKETTENVERGLEICH ERFOLGREICH BEENDET)
- Lemez összehasonlítás OK.

DISKCOMP. Az összehasonlított lemeztartalmak megegyeznek.

DISKETTE IS NOT A BACKUP DISKETTE
(DISKETTE IST KEINE SICHERUNGSDISKETTE)
- A disc nem tartalék disc

BACKUP és RESTORE. A lemez nem BACKUP programmal lett előállítva. A tartalék (backup) lemezen az első file-azonosító a BACKUPID.@@@. ismételjük meg a végrehajtást a megfelelő lemezzel.

DIVIDE OVERFLOW
(TEILERÜBERLAUF)
- Túlcsordulás az osztás műveletben

DOS. A programban 0-val való osztás fordult elő, vagy olyan belső hibát tartalmaz, amely a program megszakításához vezetett. A rendszer programmegszakítást szimulál.

DOES <út> SPECIFY A FILENAME OR DIRECTORY NAME ON THE TARGET (F = FILE, D = DIRECTORY)?
- Az <út>-ról el kell dönteni, hogy file-név, vagy altartalomjegyzék neve (F = file-név, D = altartalomjegyzék).

XCOM. Az üzenet az <út>-ról nem tudja eldönteni, hogy file-nevet, vagy altartalomjegyzéket definiál. Ha D-vel válaszolunk, akkor az XCOPY a cél lemezen létrehozza az adott nevű altartalomjegyzéket.

DO YOU SEE THE LEFTMOST 0 (Y/N)?
(IST DIE GANZ LINKS STEHENDE 0 ZU SEHEN (J/N)?)
- Látja a legbaloldalibb 0-t (I/N)?

MODE. A MODE parancsot a /R /T paraméterrel adtuk meg. A rendszerüzenet addig ismétlődik, amíg Y-t, ill. J-t nem adunk.

DO YOU SEE THE RIGHMOST 9 (Y/N)?
(IST DIE GANZ RECHTS STEHENDE 9 ZU SEHEN (J/N)?)
- Látja a legjobboldalibb 9-t (I/N)?

MODE. A MODE parancsot a /L /T parraméterrel adtuk meg. A rendszerüzenet addig ismétlődik, amíg Y-t, ill. J-t nem adunk.

DO YOU WISH TO USE THE ENTIRE FIXED DISC FOR DOS (Y/N)...?[d]
DO YOU WISH TO USE THE MAXIMUM SIZE FOR A DOS PARTITION AND MAKE THE DOS PARTITION ACTIVE (Y/N)...[]
(SOLL DIE GESAMTE FESTPLATTE FÜR DOS BENUTZT WERDEN (J/N)...?[d]
- A teljes fix lemezt a DOS számára foglalja le (I/N)...?[d]

FDISK. A DOS partíció előállítása lehetőséget választottuk és a fix lemez nem volt még felosztva, akkor jelenik meg ez a kérdés. Ha N-t adunk meg, akkor meg kell adni az előállítandó DOS partíció határait is.

COMMAND FORMAT: DISKCOPY d: d: [/1]
- A parancs alakja: DISKCOPY d: d: [/1]

DISKCOPY. A program érvénytelen paramétert talált, s válaszul kijelezte a parancs helyes alakját.

DOS PARTITION ALREADY EXISTS
- A DOS partíció már létezik

FDISK. A fix lemezegység partíciós táblója már tartalmaz egy DOS partíciót. Mégegyet nem hozhatunk létre.

DOS PARTITION CREATED
- A DOS partíciót létrehozta

FDISK. Információs üzenet: a parancs a fix lemezen létrehozta a DOS partíciót.

DOS PARTITION DELETED
- A DOS partíciót törölte

FDISK. Információs üzenet: a fix lemezen már nincs DOS partíció.

DRIVE d: ALREADY DELETED
- a d: egységet már töröltük.

FDISK. Információs üzenet: a képernyőn a parancs jelzi, hogy melyik logikai egységet töröltük már.

DRIVE LETTER MUST BE SPECIFIED
- Az egység betűjelét meg kell adni

FORMAT. A parancs a 3.20-as verziótól nem használható a lemezegység azonosítójának megadása nélkül.

DRIVE LETTERS HAVE BEEN CHANGED OR DELETED
- a logikai egységekhez rendelt betűjelek megváltoztak, vagy törlődtek.

FDISK. A kibővített DOS partícióból töröltünk egy, vagy több logikai egységet. Ennek hatására a meglevők módosultak.

DRIVE TYPES OR DISKETTE TYPES NOT COMPATIBLE
- Az egységek vagy lemezek típusa eltérő

DISKCOPY vagy DISKCOMP. A cél és forrás lemez típusa öszeférhetetlen.

DUP RECORD TOO COMPLEX
- A DUP rekord túl bonyolult

LINK. A DUP rekord túl bonyolult ahhoz, hogy a LINK fel tudja dolgozni. Egyetlen DUP 1024 byte-ot foglal le a kiterjesztés előtt. A DUP-ot tartalmazó sort egyszerűsíteni kell.

DUPLICATE FILENAME OR FILE NOT FOUND
(DOPPELTER DATEINAME ODER DATEI NICHT GEFUNDEN)
- Duplikált vagy nem található file

RENAME. Egy olyan azonosítót akartunk adni a file-nak, amely már létezik, vagy az átnevezendő adatállomány nem található a lemezen.

E...

ENTER CURRENT VOLUME LABEL FOR DRIVE d: (PRESS ENTER FOR NONE):
- Kérem a d: egység kötetazonosítóját: (< ENTER > ha nincs)

FORMAT. A 3.20-as verziótól a fix lemezegység csak akkor formázható meg, ha beírjuk a címkéjét. Ha ez nem jó, a parancs nem hajtódik végre.

ENTER THE NUMBER OF THE PARTITION YOU WANT TO MAKE ACTIVE...:[]
(GEBEN SIE DIE ZU AKTIVIERENDE PARTITION EIN...[])
- Adja meg az aktivizálandó partíciót...:[]

FDISK. Az aktív partíció változtatásának lehetőségét kértük, ezért most meg kell adni az aktivizálandó partíció számát (azonosítóját). Az azonosító a rendszerüzenetben visszajelzésre kerül, majd az < ENTER > billentyűt kell lenyomni.

ENTER PARTITON SIZE...:[DDDD]
(AUSGABE DER PARTITIONSGRÖBE...:[DDDD])
- Adja meg a partíció méretét...:[DDDD]

FDISK. A DOS partíció beállítása során jelenik meg ez az üzenet. A DOS partíció méretét kell megadni sávokban. A szögletes zárójelek között megjelent szám a standard értéket jelöli. Ha csak az üzenet végét jelző billentyűt nyomjuk le, akkor ez lesz a partíció mérete.

ENTER PRIMARY FILE NAME
(EINGABE DES PRIMÄREN DATEINAMENS)
- Adja meg a file-azonosítót

COMP. Az első file azonosítójának kérése.

ENTER 2ND FILE NAME OR DRIVE ID
- Adja meg a második file nevét vagy a lemezegységet

COMP. A második file azonosítójának kérése.

ENTER STARTING CYLINDER NUMBER...:[dddd]
(STARTZYLINDER ANGEBEN...:[dddd])
- Adja meg a kezdő cylinder számát...:[dddd]

FDISK. A DOS partíció előállítása lehetőség felkéri a felhasználót, hogy adja meg a partíció kezdetének cylinder számát. A szögletes zárójelben a legelső lehetséges értéket adja meg a program, egyébként a cylinder kezdő címét kell beadni.

ENTRY ERROR (EINGABEFEHLER)
- Parancs megadási hiba

EDLIN. Az utolsó parancsban található szintaktikai hibát javítani kell.

ENTRY HAS A BAD ATTRIBUTE OR SIZE OR LINK
(EINTRAG HAT FALSCHES ATTRIBUT (GRÖBE ODER VERBINDUNG))
- A belépési attributum rossz (méret vagy kapcsolat)

CHKDSK. Az üzenet egy vagy két ponttal kezdődhet. Azt adja meg, hogy az altartalombejegyzés rossz. A /F esetén a CHKDSK megpróbálja javítani a hibát.

EOF MARK NOT FOUND
(EOF MARKIERUNG NICHT GEFUNDEN)
- Az EOF jel nem található

COMP. A file-(ok) utolsó blokkjában nem sikerült az érvényes adatok végét megtalálni. Valószínű, hogy nem szöveges file-okat hasonlítottunk össze. A részletes információ a COMP parancsnál megtalálható.

ERROR DURING READ OF FONT FILE
- hiba a font-file olvasása közben.

MODE. A PREPARE alparancs végrehajtása során a font-file olvasása közben hiba történt.

ERROR FOUND, F PARAMETER NOT SPECIFIED. CORRECTIONS WILL NOT BE WRITTEN TO DISK
(FEHLER, PARAMETER F NICHT ANGEGEBEN. KORREKTUREN WERDEN NICHT AUF PLATTE GESCHRIEBEN)
- Hiba fordult elő, az F paraméter nem volt megadva, a javítást nem fogja a lemezre írni

CHKDSK. A /F paramétert nem használtuk. A CHKDSK az analízist úgy hajtja végre, hogy a felfedezett hibákat átlépi, így a felhasználó az analízis eredményeit láthatja. A javítások nem íródnak a lemezre.

ERROR IN COUNTRY COMMAND
- hiba a COUNTRY parancsban.

DOS. A COUNTRY parancsban megadott forma nem megfelelő, vagy a megadott file inkorrekt információt tartalmaz.

ERROR IN EXE FILE
(FEHLER IN EXE-DATEI)
- Hiba az EXE file-ban

DOS. A szerkesztett program megváltozott, ezért néhány eltolási-érték (displacement) hibás.

ERROR IN EXE/HEX FILE (FEHLER IN EXE/HEX DATEI)
- Hiba az EXE/HEX file-ban

DEBUG. A file érvénytelen jeleket tartalmaz.

ERROR LOADING OPERATING SYSTEM
(FEHLER BEI LADEN DES BETRIEBSSYSTEMS)
- Hiba az operációs rendszer betöltésének folyamán

START. A fix lemezről való töltés során hiba keletkezett. Ha a hiba ismételten előfordul, akkor a rendszert a tartalék DOS lemezről kell betölteni, és a SYS paranccsal az új másolatát kell létrehozni a fix lemezen.

ERROR OPENING LOG FILE
- hiba a LOG-file megnyitása közben.

BACKUP. A /L opciót használtuk, de a LOG-file létrehozása, vagy megnyitása közben hiba történt.

ERROR READING FIXED DISK (FEHLER BEIM LESEN DER FESTPLATTE)
- Hiba a fix lemez olvasása során

FDISK. A program az aktuális fix lemez kezdeti betöltő rekordját öt kísérlettel sem tudta beolvasni. Meg kell kísérelni az FDISK ismetelt végrehajtását. Ha ez nem sikerül, akkor kapcsolatot kell keresni a PC szállítójával.

ERROR WRITING FIXED DISK
(FEHLER BEIM SCHREIBEN AUF FESTPLATTE)
- Hiba a fix lemezre íráskor

FDISK. A program az aktuális fix lemez kezdeti betöltő rekordját öt kísérlettel sem tudta felírni. Meg kell kísérelni az FDISK ismételt végrehajtását. Ha ez nem sikerül, akkor kapcsolatot kell keresni a PC szállítójával.

ERROR WRITING TO DEVICE
(FEHLER BEI SCHREIBEN AUF EINHEIT)
- Hiba az egységre íráskor

PARANCSOK. A DOS a megadott számú byte-ot nem tudta az egységre írni. A felhasználó több adatot akart átküldeni a lemezre, mint amennyit az várt.

ERRORS ON LIST DEVICE INDICATE THAT IT MAY BE OFF-LINE. PLEASE CHECK
(DIE AUSGABEEINHEIT IST EVENTUELL AUSGESCHALTET ODER OFFLINE. BITTE ÜBERPRÜFEN)
- A listázó egység kikapcsolt (OFF line) állapotban van. Kérem ellenőrizni

PRINT. A háttér nyomtató nincs bekapcsolva. Akkor kapjuk ezt az üzenetet, ha ki van kapcsolva az egység, és új PRINT parancsot adtunk ki.

ERROR READING PARTITION TABLE
- Partíciós tábla olvasási hibás

FORMAT. A fix lemez partíciós táblája nem olvasható, valószínűleg fizikailag hibás. Próbáljuk meg az FDISK programot újra futtatni.

ERROR WRITING PARTITION TABLE
- A partíciós tábla írási hibás

FORMAT. A fix lemez partíciós táblája nem írható, valószínűleg fizikailag hibás. Próbáljuk meg az FDISK programot újra futtatni.

EXE AND HEX FILES CANNOT BE WRITTEN
(EXE UND HEX-DATEINEN KÖNNEN NICHT GESCHRIEBEN WERDEN)
- Az EXE és a HEX file-okra nem lehet írni

DEBUG. A DEBUG nem támogatja .EXE és .HEX kiterjesztésű file-ok kiírását. Más névvel írjuk ki a file-t.

EXTENDED DOS PARTITION ALREADY EXISTS
- a kiterjesztett DOS partíció már létezik.

FDISK. A fix-lemezen egynél több kiterjesztett DOS partíciót nem tudunk létrehozni.

EXTENDED DOS PARTITION CREATED
- a kiterjesztett DOS partíció létrehozása megtörtént.

FDISK. A fix-lemezen létrehoztunk egy kiterjesztett DOS partíciót. Ennek használatához logigai egységeket kell definiálnunk.

EXTENDED DOS PARTITION DELETED
- a kiterjesztett DOS partíció törölve.

FDISK. Információs üzenet.

EXTENDER CARD SWITCHES DO NOT MATCH THE SYSTEM MEMORY SIZE
- A memória bővítő kártya kapcsolói rosszul tükrözik a memória méretét

VDISK. A memóriabővítő kártya mikrokapcsolóit nem megfelelően állítottuk be, és nem jól tükrözi a memória nagyságát.

EXEC FAILURE (FEHLER BEI EXEC)
- Végrehajtási hiba

PARANCSOK. A parancs végrehajtása során hiba fordul elő, vagy a FILES = xx értéknek a konfigurációs file-ban (CONFIG.SYS) túl kicsi értéket adtunk meg. Ezt az értéket meg kell növelni, és a DOS-t ismételten be kell tölteni.

F...

FAILURE TO ACCESS CODE PAGE FONT FILE
- a kód-lap font-file-jának sikertelen elérése.

MODE. A PREPARE alparancs nem tudta a font-file-t használni. Vagy rosszul írtuk be a file nevét, vagy megsérült a file.

FAILURE TO ACCESS COUNTRY.SYS
- a COUNTRY.SYS file sikertelen elérési kísérlete.

SELECT. A COUNTRY.SYS file megnyitása sikertelen, ezért a SELECT parancs abortál.

FAILURE TO ACCESS DEVICE: xxx
- az xxx egység elérése sikertelen.

MODE. Valamelyik kód-laphoz tartozó művelet végrehajtása során az xxx egységen hiba történt. Ellenőrizzük az egység nevét, vagy a CONFIG.SYS file-ban a megfelelő DEVICE parancsot. Ezután újra adjuk ki a MODE parancsot.

FAILURE TO ACCESS KEYBOARD.SYS
- a KEYBOARD.SYS file elérése sikertelen.

SELECT. A parancs nem tudta megnyitni a KEYBOARD.SYS file-t, ezért abortált. Ellenőrizzük, hogy a file rendben a kijelölt úton található-e.

FASTOPEN ALREADY INSTALLED
- a FASTOPEN program már a memóriában van.

FASTOPEN. A parancs tranziens részét másodszor is a memóriába akartuk tölteni. Ez lehetetlen, helyette a fenti üzenetet kapjuk.

FASTOPEN INSTALLED
- a FASTOPEN programot betöltöttük.

FASTOPEN. Információs üzenet.

FATAL: INTERNAL STACK FAILURE, SYSTEM HALTED
- Fatális hiba: nincs hely a veremben, a rendszer leállt

A rendszer üzemelése közben nem volt elég hely a veremben. Ennek oka: feltehetően gyorsan ismetlődő hardware megszakítások, amelyeket a rendszer már nem tud kezelni. Tipikusan ilyen hibát okozhat egy rossz billentyűzet. Ha mégis szükség van a megszakításokra (pl. egy analóg mérőkártya esetén), akkor módosítsuk a verem méretet a STACKS konfigurációs paranccsal.

FILE ALLOCATION TABLE BAD, DRIVE X ABORT, RETRY, IGNORE?
(FEHLERHAFTE DATEIZUORDNUNGSTABELLE, LAUFWERK X ABBRUCH, WIEDERHOLEN, IGNOIEREN?)
- Az x meghajtón lévő lemez file-allokációs táblázata (FAT) rossz, A, R, I?

DOS. Az első felhasználói válasz: R ill. W. Ha a hiba ismételten előfordul, a lemez használhatatlan, újra kell formázni.

<FILE> AND <FILE>
(<FILE-SPEC> AND <FILE-SPEC>)
- <file-specifikáció> és <file-specifikáció>

COMP. Ha a COMP parancsot teljes tartalomjegyzékre (altartalomjegyzék) adjuk ki, akkor az összehasonlítás előtt a két file specifikációját (meghajtó, út) is kiírja.

FILE xxx CANCELED BY OPERATOR
(DATEI VON BEDIENER GESTRICHEN)
- A file-t törölte az operátor

PRINT. Ez az üzenet a nyomtatóra kerül, miután a nyomtatás megszakadt. Jelzi, hogy a nyomtatás nem teljes.

FILE CANNOT BE CONVERTED
- A file-t nem lehet konvertálni

EXE2BIN. A konvertálni kívánt file struktúrája nem megfelelő, ezért a .COM file-t a parancs nem tudja előállítani.

FILE CANNOT BE COPIED ONTO ITSELF
(DATEI KANN NICHT IN SICH SELBST KOPIERT WERDEN)
- A file nem másolható önmagára

DOS. Egy file-t másolni akarunk. A másolatot ugyanazzal a névvel ugyanabba a tartalomjegyzékbe akarjuk tölteni az eredeti lemezen. Ebben az esetben a másolat nevét meg kell változtatni, és/vagy más tartalomjegyzékbe vagy lemezre kell másolni.

FILE CREATION ERROR
(FEHLER BEIM ERSTELLEN DER DATEI)
- File előállítási hiba

DOS és PARANCSOK. Nem sikerült a tartalomjegyzékben új file-nevet elhelyezni, vagy a már meglevő file-t felülírni. Ha már létező file-ról van szó, akkor az csak olvasásra érhető el, ezért nem lehetett felülírni. Különben CHKDSK parancsot kell végrehajtani, hogy megállapítsuk, tele van-e a tartalomjegyzék, vagy más körülmény vezetett-e a hibához.

FILE IS CROSS-LINKED : ON CLUSTER XX
(DATEI HAT QUERVERWEIS IM BEREICH XX)
- A file fordított kapcsolást tartalmaz

CHKDSK. Ezt az üzenetet mindig kétszer kapjuk meg hibás láncolás esetén. Ekkor mindkét file új azonosítót kap. Mindkét file-hoz ugyanaz az adatblokk rendelődik. A hiba kiküszöbölése nem automatikus. A felhasználó ezt a következőképpen teheti.

- Mindkét file másolása COPY paranccsal
- Az eredeti file törlése az ERASE paranccsal
- A file-ok tartalmának javítása és szerkesztése

< FILE > IS CURRENTLY BEING PRINTED / < FILE > IS IN QUEUE
(DATEI WIRD GEDRUCKT / DATEI STEHT IN WARTESCHLANGE)
- A file nyomtatóra került / a file várakozási sorban van

PRINT. Ezt az üzenetet akkor kapjuk, ha a PRINT parancsot paraméterrel adtuk meg. (Egy vagy több file-azonosítót a várakozási sorba /queue/ helyeztünk.) Ez az üzenet a felhasználó informálását szolgálja.

FILE IS READ-ONLY
- A file csak olvasható

EDLIN. A megadott file csak olvasható.

FILENAME MUST BE SPECIFIED
- A file-nevet meg kell adni

EDLIN. Az EDLIN parancsból hiányzik a szerkesztendő file neve.

FILE NOT FOUND
(DATEI NICHT GEFUNDEN)
- A file-t nem találja

DOS és PARANCSOK. A file azonosítója a megadott tartalomjegyzékben nem szerepel.

FILE NOT IN PRINT QUEUE
- A file nincs a várakozási sorban

PRINT. A törölni kívánt file nincs a várakozási sorban.

FILE SHARING CONFLICT
- File megosztási hiba

COMP. A parancs nem tudja összehasonlítani a file-okat, mert egyikük osztott és egy másik processz használja.

FILES ARE DIFFERENT SIZES
(DATEINEN HABEN UNTERSCHIEDLICHE GRÖBEN
- A file-ok különböző méretűek

COMP. Az összehasonlítandó file-ok mérete különböző. Az összehasonlítást nem végzi el.

FILES COMPARE OK
(DATEIENVERGLEICH OK)
- File-ok összehasonlítása OK

COMP. Az összehasonlított két file azonos információkat tartalmaz.

FILES WERE BACKED UP XX/XX/XX
- A file-okat xx/xx/xx-kor mentettük

RESTORE. Tájékoztatás a mentés dátumáról.

FIRST CLUSTER NUMBER IS INVALID, ENTRY TRUNCATED
(ERSTE BEREICHSNUMBER UNGÜLTIG, EINGABE ABGESCHNITTEN)
- Az első területszám érvénytelen, a parancs beadása befejeződött

CHKDSK. A megadott file érvénytelen mutatót tartalmaz. A file 0 hosszúságúra lesz levágva.

FIXED BACKUP DEVICE d: IS FULL
- A mentésre használt fix lemez megtelt

BACKUP. A mentés fix lemezegységre történt, ami megtelt. A BACKUP parancs befejezi a mentést.

FIXUP OFFSET EXCEEDS FILELD WITH
(KORREKTUROFFSET ÜBERSCHREITET DIE FELDLÄNGE)
- A megadott offset túllépi a mező szélességét

LINK. Az assembler utasítás egy NEAR attributummal rendelkező címre vonatkozik egy FAR attributummal rendelkező helyett. Az assembler forrásprogramot javítani kell, majd újra kell fordítani.

FIXUPS NEEDED - BASE SEGMENT /HEX/:
(KORREKTUR ERFODERLICH - BASISSEGMENT /HEX/)
- Javítás szükséges. A bázis szegmens /HEX/:

EXE2BIN. A forrás (input) file (.EXE kiterjesztés) olyan információkat tartalmaz, amelyekkel azt adjuk meg, hogy a file-t szegmensenként kell tárolni. Meg kell adni az abszolút szegmenscímet, amelyre a modult betöltjük.

Figyelmeztetés!

Mivel a szegmenst egy meghatározott címre kell tölteni, ezért az ilyet xxxxxxxx.COM file-ként kell meghatározni.

FONT FILE CONTENTS INVALID
- a font-file tartalma rossz.

MODE. A font-file feldolgozása során a parancs érzékelte, hogy rossz file-t olvasott be. Lehet, hogy nem font-file-t adtunk meg, vagy a file megsérült.

FOR CANNOT BE NESTED
(FOR KANN NICHT VERSCHACHTELT WERDEN)
- A FOR parancsokat nem szabad egymásba skatulyázni

PARANCS FILE-OK. A parancs file egy sorában több FOR alparancs is található. Egy sorban csak egy FOR megengedett.

FORMAT FAILURE
(FORMATIERFEHLER)
- Formázási hiba

FORMAT. A céllemez előállítása során lemezhibát találtunk. A lemez használhatatlan.

FORMAT NOT SUPPORTED ON DRIVE d:
- A d: egységet nem lehet megformázni

FORMAT. A lemezegység kezelő program (driver) nem, vagy rosszul támogatja az általános IOCTL hívásokat. A VDISK memória lemezei nem formázhatók meg.

FORMATTING WHILE COPYING
(FORMATIEREN WÄHREND KOPIEREN)
- Másolás alatti formázás

DISKCOPY. A céllemez nem formázott. A DISKCOPY a formázatlan területet az átvitel folyamán formázza. Ha ezt az üzenetet nem megfelelő meghajtó típus üzenet követi, akkor ez azt jelöli, hogy a kétoldalú lemezt egyoldalú meghajtón akartuk másolni. A feldogozás befejeződik. A céllemez hibás adatokat tartalmaz.

G...

GRAPHIC CHARACTERS LOADED
(GRAFIKZEICHEN GELADEN)
- A grafikus karaktereket betöltötte

GRATABL. A GRAFTABL.COM file-t betöltötte.

H...

HARDWARE CODE PAGES:
PREPARED CODE PAGES:
- hardware és betöltött kód-lapok.

MODE. A parancs segítségével a kód-lapokra vonatkozó információkat kértük a képernyőre. A fenti fejlécek kiírása után a parancs kilistázza a megfelelő értékeket.

HAS INVALID CLUSTER, FILE TRUNCATED
- Hibás blokk, a file csonkolt lesz

CHKDSK. Az üzenet előtt kiírt nevű file érvénytelen mutatót tartalmaz. A /F paraméter használatával meg kell kísérelni a végét csonkolni.

HEAD : h CYLINDER: c
- Fej:h CILINDER: c

FORMAT. Az üzenettel nyomon lehet követni, hogy hol tart a lemez megformázása. Nem hibaüzenet.

I...

ILLEGAL DEVICE NAME (UNGÜLTIGER EINHEITENNAME)
- Helytelen periféria azonosító

MODE. A parancsban megadott paraméterek a következők lehetnek: nyomtató: LPT1, LPT2,LPT3; aszinkron átviteli vonal-paraméter: COM1, COM2; a MODE és a paraméterek között legfeljebb egy üres jel állhat.

INCOMPATIBLE DISKETTE OR DRIVE TYPES (UNVERTRÄGLICHE DISKETTEN-ODER LAUFWERKART)
- Nem kompatibilis lemez vagy meghajtó típus

DISKCOMP. Az első lemez mindkét oldalát sikerült elolvasni. A második lemeznek viszont csak a második oldalát sikerült. Ebben az esetben vagy a második lemez, vagy a meghajtó csak egyoldalas, vagy az első lemez sávonként 9, a második sávonként 8 szektort tartalmaz.

INCOMPATIBLE DRIVE TYPES (UNVERTRÄGLICHE LAUFWERKARTEN)
- Nem kompatibilis meghajtó típus

DISKCOMP. Az első lemez és meghajtó kétoldalas, a második lemezt tartalmazó viszont csak egyoldalas.

INCOMPATIBLE SYSTEM SIZE (UNVERTRÄGLICHE SYSTEMGRÖBE)
- Nem elfogadható rendszerméret

SYS. A céllemez olyan DOS rendszert tartalmaz, amely kisebb, mint az éppen másolandó lemez. A rendszer átvitele nem hajtható végre. Lehetséges megoldás: a FORMAT/S parancs segítségével egy üres lemezt formázunk, és a file-okat az új lemezre másoljuk.

INCORRECT APPEND VERSION
- rossz APPEND verzió.

APPEND. A program rossz verzióját használtuk. Az IBM hálózati rendszeréhez tartozó APPEND a DOS-szal nem képes együtt dolgozni.

INCORRECT DOS VERSION
(FALSCHE DOS-VERSION)
- Helytelen DOS verzió

PARANCSOK. A felhasználó által kiadott DOS parancs a DOS egy másik verzióját használja. Töltsük be a DOS megfelelő verzióját, vagy használjuk a parancs megfelelő verzióját.

INCORRECT DOS VERSION, USE DOS 2.00 OR LATER
- Rossz DOS verzió, csak 2.00 vagy újabb használható

LINK. A LINK a DOS 2.00-nál korábbi verzióival nem fut. Használjuk a megfelelő verziót!

INCORRECT NUMBERS OF PARAMETERS
- A paraméterek száma rossz

A legtöbb parancs küldheti ezt az üzenetet: a paraméterek száma nem megfelelő.

INCORRECT PARAMETER
- Rossz paraméter

SHARE. A paracsban megadott valamelyik paraméter rossz.

INFINITE RETRY ON PARALLEL PRINTER TIME-OUT
- A nyomtatón végtelen sokszor próbálkozik a rendszer

MODE. Tájékoztató üzenet: a P paramétert adtuk meg, ami lehetővé teszi a végtelen ideig történő próbálkozást.

INFINITE RETRY NOT SUPPORTED ON NETWORK PRINTER
- Hálózati nyomtatón nem lehet végtelen sokszor próbálkozni

MODE. A P paramétert használtuk. Ez nem megengedett, mert a hálózaton implementált nyomtató hibája nem kérdezhető le.

INSERT BACKUP DISKETTE XX IN DRIVE D STRIKE ANY KEY WHEN READY
(SICHERUNGSDISKETTE XX EINLEGEN IN LAUFWERK, WENN BEREIT, EINE TASTE BETÄTIGEN)
- Helyezze az xx háttér lemezt a meghajtóba, ha kész, nyomjon meg egy billentyűt

RESTORE. A következő mentett lemez behelyezése után a RESTORE a feldolgozást akkor folytatja, ha egy tetszőleges billentyűt lenyomunk.

INSERT BACKUP DISKETTE XX IN DRIVE D. WARNING! DISKETTE FILES WILL BE ERASED. STRIKE ANY KEY WHEN READY
(SICHERUNGSDISKETTE XX EINLEGEN IN LAUFWERK. ACHTUNG! DATEN WERDEN ZERSTÖRT. WENN BEREIT, EINE TASTE BETÄTIGEN)
- Az xx lemezt helyezze a meghajtóba. Figyelem! Az adatok törlődnek. Ha kész, nyomjon le egy billentyűt.

BACKUP. A következő mentési lemezt kell elhelyezni a meghajtóban. Erre csak DOS által formázott lemez használható. Egy tetszőleges billentyű lenyomásával folytatható a mentés.

INSERT BACKUP SOURCE DISKETTE IN DRIVE D STRIKE ANY KEY WHEN READY

- Helyezze be a mentendő lemezt az egységbe, utána nyomjon meg egy billentyűt

BACKUP. A rendszer a mentendő lemezt kéri. Behelyezés után egy tetszőleges billentyűt kell megnyomni.

INSERT BACKUP TARGET DISKETTE IN DRIVE D STRIKE ANY KEY WHEN READY

- Helyezze be a cél lemezt az egységbe, utána nyomjon meg egy billentyűt

BACKUP. A rendszer a cél mentési lemezt kéri. Behelyezés után egy tetszőleges billentyűt meg kell nyomni.

INSERT COMMAND.COM DISK IN DRIVE D AND STRIKE ANY KEY, WHEN READY

(COMMAND.COM EINLEGEN IN LAUFWERK D WENN BEREIT, EINE BELIEBIGE TASTE BETÄTIGEN)

- Helyezze a COMMAND.COM-ot tartalmazó lemezt a meghajtóba, ha kész, nyomjon le egy tetszőleges billentyűt

DOS. A DOS megkísérli a parancsprocesszor (COMMAND.COM) ismételt betöltését, de a parancsprocesszor nem található azon a meghajtón, amelyről a DOS indult. A DOS lemezt be kell tenni a meghajtóba, majd egy tetszőleges billentyűt le kell nyomni.

INSERT DISK TO BE RECOVERED INTO DRIVE X: AND PRESS ANY KEY WHEN READY

(WIEDERHERZUSSTELLENDE DISKETTE EINLEGEN IN LAUFWERK D WENN BEREIT, EINE TASTE BETÄTIGEN)

- Tegye a meghajtóba a visszaállítandó lemezt, majd nyomjon meg egy tetszőleges billentyűt

RECOVER. INSERT DISKETTE FOR DRIVE D AND PRESS ANY KEY WHEN READY

- Helyezzen be egy lemezt az D meghajtóba, utána nyomjon meg egy billentyűt

DOS. Logikai lemezek használatakor a rendszer a meghajtóba kér egy lemezt. Ha nem a kívánt lemez van a meghajtóban, helyezzük be a megfelelőt, majd nyomjuk meg akármelyik billentyűt.

INSERT DISK WITH BATCH FILE AND STRIKE ANY KEY WHEN READY

(DISKETTE MIT DER STAPELDATEI EINLEGEN. WENN BEREIT, EINE TASTE BETÄTIGEN)

- A parancs-file-t helyezze a meghajtóba, ha kész, nyomjon le egy billentyűt

DOS. A parancs-file-t tartalmazó lemezt kivettük. A parancsprocesszor a következő parancsot keresi a file-ban. Ha a lemezt a helyes meghajtóba visszatesszük, majd egy tetszőleges billentyűt lenyomunk, folytatódik a feldolgozás.

INSERT DISKETTE FOR DRIVE x AND PRESS ANY KEY WHEN READY:

- helyezzen az x egységbe egy lemezt, majd nyomjon meg egy tetszőleges billentyűt.

DOS. A megjelölt lemezegység logikai egység, s az adott egységben ettől eltérő van. A megfelelő lemez behelyezése után nyomjunk meg egy tetszőleges billentyűt.

INSERT DOS DISK IN D AND STRIKE ANY KEY WHEN READY

(DOS-DISKETTE EINLEGEN IN LAUFWERK D WENN BEREIT, EINE TASTE BETÄTIGEN)

- A DOS lemezt helyezze a meghajtóba, ha kész, nyomjon meg egy billentyűt.

SYS és FORMAT. A FORMAT vagy a SYS megpróbálta a DOS betöltését. A megadott meghajtóban nem DOS rendszerlemez van. Helyezzük be a megfelelő DOS rendszerlemezt, majd nyomjunk meg egy billentyűt.

INSERT DOS DISKETTE IN DRIVE A: PRESS ANY KEY WHEN READY

- A DOS lemezt tegye az A: meghajtóba, ha kész, nyomjon meg egy tetszőleges billentyűt

FDISK. A pillanatnyi fix lemezen a DOS partíciót sikerült előállítani. A DOS lemezt az A: meghajtóba kell tenni, majd egy tetszőleges billentyűt le kell nyomni. Ezután a PC DOS újraindul (BOOT). A DOS a pillanatnyi fix lemezhez azonosítót rendel.

INSERT FIRST DISKETTE IN DRIVE X INSERT SECOND DISKETTE IN DRIVE Y

(ERSTE DISKETTE EINLEGEN IN LAUFWERK X ZWEITE DISKETTE EINLEGEN IN LAUFWERK Y)

- Helyezze az első lemezt az X, a másodikat az Y meghajtóba

DISKCOMP. Az egyik, majd a másik összehasonlítandó lemezt a megadott meghajtókba kell tenni. Ezután a következő üzenet jelenik meg: Press any key when ready (Wenn bereit, eine Taste betätigen) ennek jelentése: ha kész, nyomjon le egy billentyűt. Ha ez megtörtént, az összehasonlítás elkezdődik.

INSERT KEYBxx.COM DISKETTE IN DRIVE D: AND STRIKE ANY KEY WHEN READY

- Helyezze be a KEYBxx.COM-ot tartalmazó lemezt az egységbe, majd nyomjon meg egy billentyűt

A rendszer töltésénél jeleztük, hogy a billentyűzet rutin külön lemezen van. Helyezze be a megfelelő lemezt, majd nyomjon meg egy billentyűt.

INSERT LAST BACKUP TARGET IN DRIVE D

- Helyezze be az utolsó mentési cél lemezt a egységbe

BACKUP. A parancs /A paraméterét használtuk. Az utolsó mentési lemez behelyezését kéri a parancs. Nem hibaüzenet.

INSERT NEW DISKETTE FOR DRIVE D AND PRESS ENTER WHEN READY
- Helyezzen be egy új lemezt az egységbe, majd nyomjon meg egy billentyűt. (DOS 3.0-tól < ENTER > -t)

FORMAT. A parancs kéri, hogy a lemezegységbe helyezzünk be egy új lemezt, amit meg akarunk formázni. Ezt követően nyomjuk meg az < ENTER > billentyűt.

INSERT SOURCE DISKETTE IN DRIVE X INSERT TARGET DISKETTE IN DRIVE Y
(QUELLENDISKETTE EINLEGEN IN LAUFTWERK Y)
- Helyezze a forráslemezt az X, a céllemezt az Y meghajtóba

DICKCOPY. A megfelelő lemezeket a megadott meghajtókba kell helyezni, és egy tetszőleges billentyűt le kell nyomni, ezzel folytatódik a másolás.

INSERT TARGET DISKETTE IN DRIVE D: STRIKE ANY KEY WHEN READY
- Helyezze be a céllemezt a meghajtóba, majd nyomjon meg egy billentyűt

DISKCOPY. A parancs akkor használja azt az üzenetet, ha egyetlen hajlékonylemezes egység van csak a rendszerben. Helyezzük be a céllemezt a meghajtóba, majd nyomjunk meg egy tetszőleges billentyűt.

INSERT RESTORE TARGET xx IN DRIVE D: STRIKE ANY KEY WHEN READY
- Helyezze be az egységbe az xx. mentett céllemezt, majd nyomjon meg egy billentyűt

RESTORE. A parancs a következő (xx sosrszámú) céllemezt kéri a d: meghajtóba. A lemez behelyezése után nyomjunk meg egy tetszőleges billentyűt.

INSERT SYSTEM DISK IN D AND STRIKE ANY KEY WHEN READY
- Helyezze be a rendszerlemezt az x egységbe, majd nyomjon meg egy billentyűt

SYS

A parancs megkísérelte betölteni a DOS file-okat, de nem sikerült. Ezért kér a lemezegységbe egy olyan lemezt, amelyik tartalmazza a rendszer-file-okat. A lemez behelyezése után nyomjunk meg egy tetszőleges billentyűt.

INSUFFICIENT DISK SPACE
(NICHT GENÜGEND PLATTENBEREICH)
- Nincs elég hely a lemezen

DOS és PARANCSOK. A lemezen nincs elég szabad hely az adott file felírására. Ebben az esetben a lemez állapotát a CHKDSK-kel ellenőrizhetjük.

INSUFFICIENT MEMORY
(NICHT GENÜGEND SPEICHER)
- Kevés a memória

PARANCSOK. A rendelkezésre álló memóriaterület nem elégséges a parancs-funkció végrehajtására. Ebben az esetben a CONFIG.SYS állományban a BUFFERS = xx értéket kisebbre kell állítani, majd a rendszert újra kell indítani és a funkciót végre lehet hajtani. Ha ismét ezt az üzenetet kapjuk, akkor a rendszerhez tartozó memória nem elégséges a parancs végrehajtásához.

INSUFFICIENT MEMORY FOR SYSTEM TRANSFER
- Nincs memória a rendszer másolásához

FORMAT vagy SYS. Nincs elég hely a memóriában a rendszer átmásolásához.

INSUFFICIENT ROOM IN ROOT DIRECTORY. ERASE FILES FROM ROOT AND REPEAT CHKDSK
(NICHT GENÜGEND PLATZ IM STAMMVERZEICHNIS / DATEINEN AUS STAMMVERZEICHNIS LÖSCHEN UND CHKDSK WIEDERHOLEN)
- Nincs elég hely a főtartalomjegyzékben, töröljünk a főtartalomjegyzékből file-okat, és ismételjük meg a CHKDSK programot

CHKDSK. A CHKDDSK-kel az elveszített adatblokkokból file-okat lehet újra visszállítani. A főtartalomjegyzék tele van, így esetleg nem mindegyik adatblokk írható fel. Az újra előállítandó file-okat a további ellenőrzés érdekében egy másik lemezre kell másolni, és az átvizsgált lemezről törölni kell. Ezután a maradék elveszett adatokat a CHKDSK-kel újra elő lehet állítani.

INSUFFICIENT SPACE ON DISK
(NICHT GENÜGEND PLATZ AUF PLATTE)
- Nincs elég hely a lemezen

DEBUG. Egy írási parancsot adtunk ki, de a kiírandó adatok számára nincs elég hely a lemezen. Cseréljük ki a lemezt egy olyanra, amely elegendő szabad területtel rendelkezik. Ezután az írási parancsot újból kiadhatjuk. Esetleg töröljünk file-okat a lemezről, de ezután a DEBUG programot ismét be kell tölteni.

INSUFFICIENT STACK SPACE
- Nincs elég hely a veremnek

LINK. Nincs elég hely a programszerkesztő futtatásához. Fel kell szabadítani további memóriarészeket, majd újra futtatni a LINK programot.

INTERMEDIATE FILE ERROR DURING PIPE
(DATAIFEHLER WÄHREND "DATENÜBERGRABE" /PIPE)
- File hiba az átvitel során

DOS. A DOS nem tudta az ideiglenes (temporális) file-okat előállítani, mivel az eredeti tartalomjegyzék tele volt, vagy a DOS az "átviteli" (pipe) állományt nem tudta lokalizálni, vagy a lemezen nincs elég hely az átviendő adatok számára. Ekkor az eredeti tartalomjegyzékből néhány bejegyzést törölni kell, majd a parancsot meg kell ismételni.

INTERNAL STACK OVERFLOW SYSTEM HALTED
- a verem túlcsordult, a rendszer megállt.

DOS. A verem - valószínűleg ismételt hardware megszakítások miatt - megtelt. A CONFIG.SYS file-ban növeljük meg a STACKS értékét.

INVALID BAUD RATE SPECIFIED
(FALSCHE ÜBERTRAGUNGSRATE ANGEGEBEN)
- Helytelen BAUD érték megadás

MODE. Az átviteli sebesség, amelyet paraméterként adtunk meg, nem 100, 150, 300, 600, 1200, 2400, 4800 vagy 9600. (Az érték az első két karakterrel is megadható.)

INVALID CHARACTERS IN VOLUME LABEL
- Érvénytelen karakterek a kötetazonosítóban

FORMAT. A lemezegység címkéjének megadásakor nem megengedet karaktereket írtunk be. A címke megadásakor nem használhatjuk a pontott (.). A címke 1-11 hosszú karaktersorozat.

INVALID CODE PAGE SPECIFIED
- érvénytelen kód-lap.

KEYB. Az adott ország-specifikus információk mellett a kód-lap nem használható.

INVALID COMMAND.COM IN DRIVE D
(UNGÜLTIGER COMMAND.COM IN LAUFWERK x)
- Rossz a COMMAND.COM a meghajtón

DOS. A parancsprocesszor (COMMAND.COM) ismételt betöltése során kiderült, hogy a COMMAND.COM-nak egy érvénytelen verziója került a lemezre. A megfelelő DOS lemezt kell betenni, majd egy tetszőleges billentyű lenyomása után folytatható a munka.

INVALID COUNTRY CODE
- Rossz ország kód

SELECT vagy COUNTRY. A COUNTRY= parancsban vagy a SELECT parancsban megadott országkód nem létezik.

INVALID COUNTRY CODE OR CODE PAGE
- érvénytelen ország kód, vagy kód-lap.

COUNTRY. A CONFIG.SYS file-ban megadott COUNTRY parancs nem megfelelő.

INVALID CURRENT DIRECTORY
- Rossz az aktuális tartalomjegyzék

CHKDSK. A parancs nem képes az aktuális tartalomjegyzék olvasására. Ennek oka a lemez azonosíthatatlan hibája.

INVALID DATE
(UNGÜLTIGES DATUM)
- Helytelen dátum

DATE. A dátumban helytelen értéket vagy elválasztójelet adtunk meg. Érvényes elválasztójelek: a slash (/) és a kötőjel (-). Írjuk be helyesen a napi dátumot!

INVALID DEVICE
(UNGÜLTIGE EINHEIT)
- Helytelen periféria

CTTY. A megadott egységazonosító a DOS számára érvénytelen.

INVALID DEVICE PARAMETERS FROM DEVICE DRIVER
- A periféria-kezelő rossz paramétereket adott meg

FORMAT. A fenntartott szektorok száma nem egész számú többszöröse a szektor/sáv értéknek. Futtassuk újra az FDISK programot, majd kíséreljük meg ismét a FORMAT végrehajtását.

INVALID DIRECTORY
(UNGÜLTIGES INHALTSVERZEICHNIS)
- Helytelen tartalomjegyzék

DOS és PARANCSOK. A megadott úton valamelyik tartalomjegyzék nem létezik.

INVALID DISK CHANGE
- A lemezt nem szabad kicserélni

A nagy kapacitású lemezegységen kicseréltük a lemezt, miközben azon nyitott file-ok voltak. Helyezzük vissza a megfelelő lemezt.

INVALID DRIVE IN SEARCH PATH
(UNGÜLTIGES LAUFWERK IN SUCHPFAD)
- Helytelen meghajtó a keresési úton

DOS. Érvénytelen meghajtó megadás a PATH parancs által megadott úton. A DOS akkor adja ezt az üzenetet, ha parancsot vagy parancsfile-t keres, és nem akkor, amikor a hibás PATH parancsot kiadtuk.

INVALID DRIVE OR FILE NAME
(UNGÜLTIGE LAUFWERKANGABE)
- Helytelen meghajtó specifikáció

DOS, PARANCSOK. A parancsban a meghajtó megadása érvénytelen.

INVALID DRIVE SPECIFICATION. SPECIFIED DRIVE DOS NOT EXIST, OR IS NON-REMOVABLE
- Rossz egység megadása. Az egység nem létezik, vagy fix.

DOS vagy valamelyik parancs. Nem létező lemezegységre hivatkoztunk, vagy fix lemezegységre adtunk ki egy csak cserélhetőre alkalmazható parancsot.

INVALID DRIVE SPECIFICATION. SOURCE AND TARGET DRIVES ARE THE SAME
- Rossz egység megadása. A forrás és a cél lemez ugyanaz.

BACKUP vagy RESTORE. A forrás és cél lemezeket ugyanazon a meghajtón akarjuk használni, ez nem megengedett.

INVALID ENVIRONMENT SIZE SPECIFIED
- Rossz környezet méret

A megadott környezet méret vagy túl nagy, vagy nem-numerikus karaktereket tartalmaz.

INVALID FILE NAME OR FILE NOT FOUND
- Rossz, vagy nem létező file-név

RENAME vagy TYPE. Olyan file-t kíséreltünk meg átnevezni, listázni, amelynek neve rossz volt, vagy nem volt az adott tartalomjegyzékben.

INVALID FORMAT FILE
(UNGÜLTIGE FORMATADATEI)
- Helytelen formájú file

LINK. A könyvtár hibát tartalmaz. Kíséreljük meg újra a szerkesztést.

INVALID KEYBOARD CODE
- Rossz billentyűzet rutin kód

SELECT. Rossz billentyűzet kódot adtunk meg.

INVALID KEYBOARD CODE SPECIFIED
- érvénytelen billentyűzet-kód.

KEYB. A parancsban rossz billentyűzet-kódot adtunk meg.

INVALID MEDIA OR TRACK 0 BAD - DISK UNUSABLE
- Fizikailag nem megfelelő lemez

FORMAT. A parancs nem tudta megformázni a lemezt. Elképzelhető, hogy a lemez 0. sávja fizikailag hibás. Ugyanezt az üzenetet kapjuk, ha a lemez és lemezegység nem kompatibilis. (Pl. egy nagy kapacitású lemezt akarunk megformázni egy 320/360 Kbyte-os egységen.)

INVALID NUMBER OF PARAMETERS
(UNGÜLTIGE ANZAHL PARAMETER)
- Érvénytelen számú paraméter

PARANCSOK. Túl sok, vagy túl kevés paraméter a parancsban.

INVALID NUMERIC PARAMETER
(UNGÜLTIGER NUMERISCHER PARAMETER)
- Helytelen numerikus paraméter

LINK. Egy szám nem-numerikus karaktert tartalmazott.

INVALID NUMERIC SWITCH SPECIFICATION. SWITCH ERROR: "s:xxx"
- Rossz kapcsoló megadása az "s:xxx" kapcsolóban

LINK. A LINK paramétereinek megadása hibás: valamelyik kapcsoló helytelen értéket tartalmaz.

INVALID OBJECT MODULE
(UNGÜLTIGES OBJEKTMODUL)
- Helytelen tárgy-modul

LINK. A tárgy (object) modul hibásan formázott vagy nem teljes. Lehet, hogy a lemez fizikailag hibás.

INVALID PARAMETER
(UNGÜLTIGER PARAMETER)
- Helytelen paraméter

DOS és PARANCSOK. Az adott parancs paraméterei közül egy vagy több érvénytelen.
A program olyan meghajtó azonosítót vár, ahol a meghajtó azonosító után kettőspont áll. Meg kell győződni arról, hogy a slash-t (/) követő jel a program (parancs) érvényes paramétere-e.

INVALID PARAMETERS
(UNGÜLTIGE PARAMETER)
- Helytelen paraméterek

MODE. Nem adtunk meg paramétert, vagy az első jel a paraméterben nem L vagy C volt, vagy az első paraméter nem 40, 80, BW40. BW80, CO40, CO80, L, R volt, vagy a képernyő adapter (színes/grafikus), amire a parancs vonatkozik, nincs a rendszerben.

INVALID PARTITON TABLE
(UNGÜLTIGE PARTITONTABELLE)
- Helytelen partíció tábla

START. A DOS-t fix lemezről indítottuk, a lemez partíció táblázataiban érvénytelen információk fordulnak elő. Ekkor a DOS-t a floppy lemezről kell indítani, majd az FDISK segítségével korrigálni kell a fix lemez partíció-információit.

INVALID PATH
(UNGÜLTIGER PFAD)
- Helytelen útmegadás

Valamelyik parancsban megadott út rossz. Vagy nem megengedett karaktereket tartalmaz, vagy hosszabb 63 karakternél.

INVALID PATH, NOT DIRECTORY OR DIRECTORY NOT EMPTY
(UNGÜLTIGER PFAD, KEIN INHALTSVERZEICHNIS ODER INHALTSVERZEINCHNIS NICHT LEER)
- Helytelen útmegadás, nem tartalomjegyzék vagy nem üres a tartalomjegyzék

RMDIR. A megadott tartalomjegyzéket nem lehet törölni, mivel az útmegadásban levő nevek érvénytelenek voltak, vagy a megadott tartalomjegyzék tartalmaz még más tartalomjegyzékre vonatkozó altartalomjegyzék bejegyzéseket, vagy file azonosítókat. A tartalomjegyzék csak akkor törölhető, ha üres. (Kivétel a . és a .. bejegyzés.)

INVALID PATH OR FILE NAME
(UNGÜLTIGER PFAD ODER DATEINAME)
- Érvénytelen út vagy file azonosító

COPY vagy ATTRIB. Nem létező file-azonosítót vagy tartalomjegyzéket adtunk meg.

INVALID PATH OR PARAMETER
- érvénytelen út, vagy paraméter.

APPEND. A parancsban a /X, /E kapcsolókat útleírásukkal együtt használtuk, ami nem megengedett.

INVALID SIGNATURE IN COUNTRY.SYS FILE
- érvénytelen fejléc a COUNTRY.SYS file-ban.

SELECT. A COUNTRY.SYS file-ban a SELECT érvénytelen fejlécet talált. Ellenőrizzük a file-t, majd ismételjük meg a parancsot.

INVALID SYNTAX ON PRINTER.SYS CODE PAGE SWITCHING DEVICE DRIVERS
- nem megfelelő szintaxis a PRINTER.SYS kód-lap-váltó használatakor.

MODE. A PRINTER.SYS-t tartalmazó DEVICE konfigurációs parancs paraméterei nem megfelelőek. Javítsuk ki a CONFIG.SYS file-t, majd indítsuk újra a DOS-t.

INVALID SUBDIRECTORY
(UNGÜLTIGES UNTERVERZEICHNIS)
- Helytelen altartalomjegyzék

CHKDSK. A megadott altartalomjegyzékben érvénytelen információk találhatók. A CHKDSK megkísérli a hibát átlépni. A hiba típusáról akkor kapunk információt, ha a /V paraméterrel hívjuk meg a CHKDSK programot.

INVALID SWITCH
(UNGÜLTIGER SCHALTER)
- Helytelen kapcsoló

LINK. A kijelzett karakter nem érvényes LINK paraméter kapcsoló.

INVALID SWITCH CHARACTER
- Rossz karakter a kapcsolóban

VDISK. A konfigurációs parancs egy slash-t (/) talált, de az ezt követő karakter nem E volt. A memória lemezt a parancs a memória alsó részében hozta létre.

INVALID TIME
(UNGÜLTIGE ZEIT)
- Érvénytelen idő megadás

TIME. Érvénytelen értéket vagy elválasztó jelet adtunk meg. Az órák, percek, másodpercek között csak kettőspont (:), a másodpercek és a századmásodpercek között a pont (.) az érvényes elválasztó jel.

INVALID VOLUME LABEL
- Rossz kötetazonosító

FORMAT. A kért címke (kötetazonosító) nem egyezett a lemezegység címkéjével. A VOL kiadásával ellenőrizhető, mi is a helyes név. Ezután újra ki lehet adni a FORMAT parancsot.

IS KEYBxx.COM ON ANOTHER DISKETTE (Y/N)?
- A KEYBxx.COM másik lemezen van (I/N)?

A KEYBxx.COM file-t nem találta a rendszer, ezért megkérdezte, hogy másik lemezen van-e. Ha rossz billentyűzet rutin kódot adtunk meg, akkor N-nel kell válaszolni, majd beírni a helyes kodot. Ha külön lemezen tartjuk, akkor Y a helyes válasz.

K...

KEYB HAS NOT BEEN INSTALLED
- a KEYB programot még nem installáltuk.

KEYB. A KEYB parancs lekérdező funkcióját akartuk használni, holott még nem is installáltuk. Töltsük be a KEYB-t, majd ismételjük meg a parancsot.

KEYBOARD ROUTINE NOT FOUND
- A billentyűzet rutin nem található meg

A billentyűzet rutint nem találta a rendszer sem az aktuális tartalomjegyzékben, sem külön lemezen.

L...

LABEL NOT FOUND
(SPRUNGZIEL NICHT GEFUNDEN)
- Az ugrási címke nem található

PARANCSFILE-OK. A GOTO parancsban olyan címke van megadva, amely a parancsfile-ban nem található.

LAST BACKUP TARGET NOT INSERTED
- Nem az utolsó mentési lemezt tettük be

BACKUP. A parancsban használtuk a /A paramétert, de a cserélhető céllemez nem az utolsó volt a mentési lemezek közül. Tegyük be az utolsó mentési lemezt.

***** LAST FILE NOT BACKED UP *****
- Az utolsó file nincs elmentve

BACKUP. A parancs fix lemezegységre mentett. Az utoljára kimentett file másolása közben megtelt a lemez. Ezért ezt a file-t a mentésből törli a parancs. A mentés befejeződik.

LINE TOO LONG
(ZEILE ZU LANG)
- A sor túl hosszú

EDLIN. Egy jelsorozat kicserélése során a 253 sorhosszúságot átléptük. A "szöveg-rész kicserélese" parancs nem hajtódik végre. A sort rövidebb jelsorozatokra kell felbontani, majd a parancsot ismét ki kell adni.

LIST OUTPUT IS NOT ASSIGNED TO A DEVICE
(LISTAUSGABE IST KEINER EINHEIT ZUGEORDNET)
- A lista output nincs egységhez hozzárendelve

PRINT. A PRINT listázó egysége nem érvényes egység. A PRINT parancsot ismét ki kell adni, de érvényes nevet kell megadni.

LOCK VIOLATION
- Zárolási hiba

XCOPY. A forrás file, vagy annak egy része zárolt. Várni kell egy kis ideig, majd újból kiadni a parancsot.

LOGGING TO FILE < file-név >
- bejegyzések a < file-név > file-ba.

BACKUP. Információs üzenet: a /L paraméterrel megadott < file-név > file-t használja a parancs LOG-file-ként.

LOGICAL DOS DRIVE CREATED, DRIVE LETTERS CHANGED OR ADDED
- új logikai DOS egység, az egységek betűjele módosítva, vagy bővítve.

FDISK. A kiterjesztett DOS partícióban új logikai egységet hoztunk létre. Ennek hatására a már meglévő egységekhez tartozó betűjelek módosulhatnak, s legalább egy további betű is 'érvényes' lesz.

INVALID SIGNATURE IN KEYBOARD.SYS FILE
- érvénytelen fejléc a KEYBOARD.SYS file-ban.

SELECT. A KEYBOARD.SYS file-ban a SELECT érvénytelen fejlécet talált. Ellenőrizzük a file-t, majd ismételjük meg a SELECT parancsot.

INVALID STACK PARAMETER
- érvénytelen STACK paraméter.

STACKS. A CONFIG.SYS file-ban szereplő konfigurációs parancs paramétereit rosszul adtuk meg

LPT#: NOT REROUTED
(LPT#: NICHT UMGELEITET)
- LPT#: nem átirányítható

MODE. A párhuzamos interface-szel rendelkező nyomtató outputját egy soros egységre átirányítottuk. A MODE paranccsal viszont a nyomtatás szélességét vagy sűrűségét változtattuk. A rendszer az átirányítást érvényteleníti.

LPT#: REROUTED TO COMn
(LPT#: UMGELEITET ZU COMn)
- LPT#: átirányítva a COMn-re

MODE. A párhuzamos interface-szel rendelkező nyomtató outputját egy soros egységre átirányítottuk. (# = 1, 2 vagy 3; n = 1 vagy 2)

LPT#: SET FOR 80
(LPT#: AUF (80 GESETZT)
- LPT érték 80

MODE. A nyomtatási szélességet úgy próbáltuk 80-ra állítani, hogy a standard formátumot követeljük meg. Ha ez nem sikeres, akkor ezt az üzenetet egy hibaüzenet követi a képernyőn. (# = 1, 2 vagy 3)

LPT#: SET FOR 132
(LPT#: AUF 132 GESETZT)
- LPT#: érték 132

MODE. A nyomtatási szélességet úgy próbáltuk 132-re állítani, hogy a sűrített nyomtatási formát követeljük meg. Ha ez nem sikeres, akkor egy hibaüzenet is megjelenik a képernyőn. (# = 1, 2 vagy 3)

M...

MAKE SURE A DISKETTE IS INSERTED INTO THE DRIVE AND THE DOOR IS CLOSED
- Győződjön meg arról, hogy van-e lemez az egységben, és az ajtó zárva van-e

DISKCOMP vagy DISKCOPY. A hajlékonylemezes egység üres, vagy nincs az ajtaja becsukva.

MAXIMUM AVAILABLE SPACE IS NNNN CYLINDERS AT CYLINDER XXXX
MAXIMUM AVALLABLE SPACE FOR PARTITION IS xxx CYLINDERS
(MAXIMAL VERFÜGBARER BEREICH IST NNNN ZYLINDER BEI ZYLINDER XXXX)
- Maximális rendelkezésre álló terület az xxxx-től nnnn darab cilinder

FDISK. A "DOS partíció előállítása" lehetőséget kértük. A DOS partíció elhelyezkedését adjuk meg ezzel. Ezeket az értékeket standardként is használhatjuk.

MAXIMUM NUMBER OF LOGICAL DOS DRIVERS INSTALLED
- maximális számú logikai egységet már installáltunk.

FDISK vagy DOS. A DOS logikai egységei A-tól Z-ig terjedő betűjelet kapnak, így legfeljebb 24 logikai egység lehet. Ennél többet próbáltunk meglétrehozni. Ha a 25.-re is szükség van, azt csak úgy tehetjük meg, hogy valamelyik előzőt töröljük.

MEMORY ALLOCATION ERROR. CANNOT LOAD COMMAND, SYSTEM HALTED
(FEHLER BEI DER SPECHERZUORDNUNG. COMMAND KANN NICHT GELADEN WERDEN, SYSTEM ANGEHALTEN)
- Memória lefoglalási hiba miatt a parancsprocesszort nem lehet betölteni, a rendszer leállt

DOS. Egy program felülírta azt a memóriaterületet, amelyen a DOS allokált memóriablokkok láncát tartja nyilván. A DOS-t ismét be kell tölteni. (boot)

MISMATCH DOS LEVEL NUMBER
- Eltérő DOS szint szám

LINK. Programhiba. A program szállítójával kell felvenni a kapcsolatot, és tisztázni a hiba okát.

MISSING FILE NAME
(FEHLENDER DATEINAME)
- Hiányzó file név

RENAME. A második file-nevet nem adtuk meg.

MISSING FROM THE FILE IS EITHER THE DEVICE ID OT THE CODE PAGE
- a file-ból vagy az egység vagy a kód-lap hiányzik.

MODE. A kód-lapok listáján megadott sorszámok vagy a használt nyomtatótípusánaak megfelelő információ hiányzik a CPI file-ból.

MISSING OPERATING SYSTEM
(FEHLENDES BETRIEBSSYSTEM)
- Nincs operációs rendszer

START. A DOS fix lemezes indítása során a betöltő eljárás megállapította, hogy a DOS partíció indítóképesként (aktív) lett megjelölve, ennek ellenére DOS rendszert nem tartalmaz. A DOS rendszert a fix lemezre kell másolni. (A FORMAT parancsot a /S paraméterrel kell megadni, majd a rendszer file-okat is a fix lemezre kell másolni.)

MODE fff CODE PAGE FUNCTION COMPLETED
- MODE: az fff kód-lapozási funkció végrehajtva.

MODE. Információs üzenet. A QUERY, PREPARE vagy SELECT alparancsok (ez fff értéke) végrehajtása befejeződött.

MUST SPECIFY COM1 OR COM2
- COM1 vagy COM2 megadása kötelező

MODE. A parancsban nem adtuk meg a COM1 vagy COM2 használatát.

MUST SPECIFY ON OR OFF
- Csak ON vagy OFF adható meg

BREAK. A parancsnak ket paramétere lehet: ON vagy OFF. Ettől eltérő paramétert adtunk meg.

MUST SPECIFY DESTINATION LINE NUMBER
(NUMMER DER ZIELZEILE IST ANZUGEBEN)
- Meg kell adni a célsorok számát

EDLIN. A MOVE vagy a COPY parancsot sorszám-azonosító nélkül adtuk ki. A parancsot helyesen ismételten ki kell adni.

MUST SPECIFY ON OR OFF
- csak ON vagy OFF adható meg.

BREAK. A parancsban az ON vagy OFF szócskától eltérő paramétert adtunk meg.

N...

NAME OF LIST DEVICE [PRN]:
(NAME DER LISTENEINHEIT [PRN]:)
- A listázó egység azonosítója [PRN]:

PRINT. Ezt a nyomtatási folyamat indítása után kapjuk. Erre egy fenntartott egység azonosítójával, vagy az üzenetvég billentyű lenyomásával kell válaszolni, és ekkor az első párhuzamos interface-szel rendelkező nyomtatóra kerül a lista.

NEAR/FAR CONFLICT
- Közeli/távoli megadási hiba

LINK. Egy COMMON változó közeli (near) és távoli (far) attributum definíciója ellentmondó. Javítsuk ki, és futtassuk újra a LINK programot.

NETWORK SUPPORT LOADED. UNABLE TO FDISK
- FDISK hálózaton nem használható

FDISK. Egy hálózaton implementált fix lemezegységre nem használhatjuk az FDISK programot, még akkor sem, ha fizikailag ugyanazon a gépen van.

NLSFUNC ALREADY INSTALLED
- az NLSFUNC programot már installáltuk.

NLSFUNC. A parancs rezidens része már a tárban van; másodszor nem töltődik be.

NO APPEND
- nincs 'APPEND' hatálya alatt semmi.

APPEND. Az APPEND parancs lekérdező funkcióját használtuk. Válaszként megtudtuk, hogy nincs egyetlen olyan út sem, amelyiket az APPEND hatására a DOS átnézne. Az APPEND; kiadása után pl. ezt a választ kapjuk.

NO CODE PAGE HAS BEEN SELECTED
- nincs kiválasztott kód-lap.

MODE. A STATUS alparancsra válaszként kapjuk ezt az üzenetet. Jelentése a szóbanforgó eszközre egyetlen kód-lapot sem választottunk ki.

NO EXTENDED DOS PARTITION TO DELETE
- nincs kiterjesztett DOS partíció, ezértnem törölhetjük.

FDISK. Az FDISK parancsot kiterjesztett DOS partíció törlésére utasítottuk, holott az nem létezik, s ezért nem is tudjuk törölni.

NO DOS PARTITION TO DELETE
(KEINE DOS PARTITON ZU LÖSCHEN)
- Nincs törölhető partíció

FDISK. A felhasználó a "DOS partíció törlése" lehetőséget választotta, de az adott fix lemezen nincs DOS partíció.

NO FILES ADDED
- Nem adott hozzá file-t

REPLACE. A /A paramétert használtuk, de a forrás file-ok már mind léteznek a cél lemezen is.

NO FILES FOUND
- Nem talált egy file-t sem

REPLACE. A forrásban kért file-ok egyikét sem találta meg a parancs a cél lemezen.

NO FILES REPLACED
- Nincs mit helyettesíteni

REPLACE. A cél lemezen egyetlen file-t sem helyettesített a parancs, mert a forrás file-ok mindegyikére azt mondtuk, hogy nem kell helyettesíteni.

NO FIXED DISKS PRESENT
(KEINE FESTPLATTE VORHANDEN)
- A fix lemezt a rendszer nem éri el

FDISK. Az FDISK programot egy olyan PC-n hajtjuk végre, amelynek:
  - nincs fix lemeze;
  - a fix lemez a bővítési egységben található, de az nincs bekapcsolva;
  - a fix lemezt rosszul installálták

NO FREE FILE HANDLERS. CANNOT START COMMAND, EXITING
(KEINE FREIEN DATEINUMMERN. COMMAND KANN NICHT GESTARTED WERDEN, BEENDET)
- Nincs szabad file-kezelő. A START nem hajtható végre, kilépés

DOS. A parancsprocesszort (COMMAND.COM) nem lehetett ismételten behívni, mivel pillanatnyilag túl sok file van megnyitva. A CONFIG.SYS adatállományban lévő FILES = xx paraméterben a file-ok számát növelni kell, majd a DOS-t újra kell indítani.

NO LOGICAL DRIVES DEFINED
- nincs logikai egység definiálva.

FDISK. A kiterjesztett DOS partícióban egyetlen logikai egységet sem hoztunk létre. Ha az allokált lemez-területet egyáltalán használni akarjuk, akkor legalább egy logikai egységet létre kell hoznunk a kiterjesztett DOS partícióban.

NO OBJECT MODULES SPECIFIED
(KEINE OBJEKTMODULE ANGEGEBEN)
- Nem adtunk meg object modult

LINK. A parancssorban nem adtunk meg tárgy (object) modult, a LINK-nek viszont szüksége van rá.

NO PARTITIONS DEFINED
- nincs egyetlen partíció sem.

FDISK. A program jelzi, hogy a fix lemezen még egyetlen partíciót sem hoztunk létre. Ha a lemezt használni szeretnénk, legalább egyet létre kell hozni.

NO PARTITIONS TO DELETE
- nincs a lemezen partíció, ezért nem is törölhetjük azokat.

FDISK. A programot partíció törlésére utasítottuk, holott nincs egyetlen partíció sem.

NO PARTITONS TO MAKE ACTIVE
(KEINE PARTITONEN ZUM AKTIVIEREN)
- Nincs aktivizálható partíció

FDISK. A felhasználó az "aktív partíció változtatása" lehetőséget választotta, de az adott fix lemezen nincs ilyen. Ezért először a "DOS partíció előállítása" lehetőséget kell választani, és csak ezután aaktivizálható ez a partíció.

NO PATH
(KEIN PFAD)
- Nincs út

PATH. Pillanatnyilag a DOS számára nincs alternatív út kijelölve, amelyen a parancsokat vagy parancs file-okat keresbetné. Ezt az üzenetet akkor kapjuk, ha azokat a megadott vagy az aktuális tartalomjegyzékben nem találja.

NO PRIMARY DOS PARTITION TO DELETE
- nincs elsődleges DOS partíció, ezért azt nem is törölhetjük.

FDISK. A programot az elsődleges DOS partíció törlésére utasítottuk, holott még létre sem hoztuk.

NO RETRY ON PARALLEL PRINTER TIME-OUT
- Időtúllépés esetén nincs ismétlés

MODE. Az üzenet tájékoztat arról, hogy a P paramétert nem adtuk meg, ezért az időtúllépés esetén a nyomtatás megszakad, és nem kísérletezik tovább.

NO ROOM FOR SYSTEM ON DESTINATION DISK
(KEIN PLATZ FÜR DAS SYSTEM AUF DER ZIELDISKETTE)
- Nincs hely a rendszer számára a céllemezen

SYS. A feltett lemezen nem található a DOS számára fenntartott terület, ezért a rendszer nem vihető át. Ekkor egy üres lemezt meg kell formázni (FORMAT /S), és így lehet a rendszert létrehozni.

NO ROOM IN DIRECTORY FOR FILE
(DAS INHALTSVERZEICHNIS IST VOLL)
- Nincs hely a tartalomjegyzékben a file számára

EDLIN. A tartalomjegyzék a megadott lemezen megtelt. A végrehajtott változtatások elvesztek. Indítsuk újra az EDLIN-t egy üres, formázott lemezre.

NO ROOM IN ROOT DIRECTORY
- Nincs hely a főtartalomjegyzékben

LABEL. A címke (kötetazonosító) létrehozása a főtartalomjegyzékben egy bejegyzést igényel. Ez azonban nem lehetséges, mert az tele van.

NO SOURCE DRIVE SPECIFIED
- a forrás egységet nem adtuk meg.

BACKUP. A forrás egységet meg kell adni.

NO SPACE FOR A XXXX CYLINDER PARTITON
(KEIN PLATZ FÜR EINE PARTITON IT XXXX ZYLINDERN)
- Nincs hely az xxxx cilinderes partíció számára

FDISK. A megadott partíció cilinderméret nagyobb, mint a szabad terület a lemezen. Ebben az esetben kisebb értéket kell megadni.

NO SPACE FOR A NNNN CYLINDER PARTITION AT CYLINDER XXXX
(KEIN PLATZ FÜR EINE PARTITION MIT NNNN ZYLINDERN BEI ZYLINDER XXXX)
- Az xxxx cilindernél lévő nnnn cilinderes partíció számára nincs hely

FDISK. Egy partíciót a fix lemez egy meghatározott helyén akartunk előállítani. Ezen a helyen viszont nincs elég szabad terület a DOS partíció számára.

NO SPACE TO CREATE A DOS PARTITON
(KEIN PLATZ FÜR DAS ERSTELLEN EINER DOS-PARTITON)
- Nincs hely a DOS partíció létrehozásához

FDISK. A "DOS partíció előállítása" lehetőséget választottuk. A fix lemezen viszont nincs több szabad hely.

NO SPACE TO CREATE LOGICAL DRIVE
- nincs hely a logikai egység létrehozására.

FDISK. A kiterjesztett DOS partícióban levő összes helyet már más logikai egységek használják, ezért továbbiakat már nem hozhatunk létre.

NO SUB-DIRECTORIES EXIST
(KEINE UNTERVERZEICHNISSE VORHANDEN)
- Altartalomjegyzékek nem léteznek

TREE. A megadott meghajtón csak eredeti tartalomjegyzék van, ezért elérési utat nem kell megadni.

NO SYSTEM ON DEFAULT DRIVE
- Az aktuális lemezen nincs rendszer

SYS. Az átmásolandó rendszer file-okat a parancs nem találta az aktuális lemezegységen.

NO TARGET DRIVE SPECIFIED
- nem adtuk meg a cél egységet.

BACKUP. A parancsban mind a forrás, mind a cél egységet meg kell adni.

NO VERSION OF GRAPHIC CHARACTER SET TABLE IS ALREADY LOADED
- a grafikus karakterkészlet semelyik verziója sincs betöltve.

GRAFTABL. A parancs jelzi, hogy nem töltötte be a grafikus karakterkészletet, mert a DOS már betöltötte. Információs üzenet.

NON-DOS DISKETTE
- Nem DOS lemez

CHKDSK vagy valamelyik parancs. A lemez formátuma felismerhetetlen.

NON-STANDARD VERSION OF GRAPHIC CHARACTER SET.TABLE IS ALREADY LOADED

GRAFTABL. Az üzenet jelzi, hogy a memóriába töltött grafikus karakterkészletet módosítottuk, ezért azt a rendszer nem tudja azonosítani. Ha a GRAFTABL segítségével egy új készletet töltünk be, akkor az a RAM-ban levő táblázatot felülírja.

NON-SYSTEM DISC OR DISC ERROR. REPLACE AND STRIKE ANY KEY WHEN READY
(KEINE-SYSTEMPLATTE ODER PLATTENFEHLER. AUSTAUSCHEN WENN BEREIT, EINE TASTE BETÄTIGEN)
- Nem rendszerlemez vagy rendszerhiba. Cseréljük ki a lemezt, majd nyomjunk le egy tetszőleges billentyűt.

START. Az xxxBIO.COM vagy az xxxDOS.COM számára nincs bejegyzés a tartalomjegyzékben, vagy a rendszer indítása során olvasási hiba fordult elő. Az A: meghajtóba DOS lemezt kell helyezni, majd a rendszert újraindítani.

***** NOT ABLE TO BACK UP FILE *****
- A file nem menthető

BACKUP. A file-t nem lehet menteni, mert osztott. Próbáljuk meg később a mentést, és használjuk a /M paramétert.

***** NOT ABLE TO RESTORE FILE *****
- A file nem tölthető vissza

RESTORE. A visszatöltendő file-t nem lehet megnyitni, mert zárolt.

NOT ENOUGH MEMORY
- Nincs elég memória

SHARE vagy REDIR. A felhasználható memória kevesebb, mint amennyi a konfigurációs parancsok végrehajtásához szükséges, ezért ezek installálására nem került sor.

NOT ENOUGH ROOM TO MERGE THE ENTIRE FILE
(NICHT GENÜGEND PLATZ ZUM EINLESENDER GESAMTEN DATEI)
- Nincs elég terület a teljes file szerkesztéséhez

EDLIN. Az átviteli parancs a megadott file teljes tartalmát nem tudta beolvasni a memóriába, mivel nem állt rendelkezésére elég szabad terület. A file-nak csak egy részét olvasta be.

NOT FOUND
(NICHT GEFOUND)
- Nem található

EDLIN. A megadott sor-intervallum nem tartalmazza azt a jelsorozatot, amit a "szöveg keresése" parancsban megadtunk. Előfordulhat például, hogy az OK üzenetre N-nel válaszoltunk, de a megadott jelsorozat többször már nem fordult elő a szövegben.

O...

ONE OR MORE CON CODE PAGES INVALID FOR GIVEN LANGUAGE
- néhány CON kód-lap érvénytelen az adott nyelv esetén.

KEYB. A KEYB parancs a használt kód-lapokhoz további táblázatokat hoz létre. Ha valamelyik kód-lap sorszáma érvénytelen az adott nyelv esetén, akkor ezt nem tudja létrehozni. Az érvényes kód-lapokban ezt meg tudja tenni, azok használhatók.

ONLY NON-BOOTABLE PARTITIONS EXIST
- csak nem tölthető partíciók léteznek.

FDISK. A programot az aktív partíció módosítására utasítottuk, holott a fix lemezen egyáltalán nem volt aktív partíció.

ONLY PARTITIONS ON DRIVE 1 CAN BE MADE ACTIVE
- csak az első fix egységen hozható létre aktív partíció.

FDISK. Operációs rendszer csak az első fix lemezről tölthető. Ezért a többi fix egységen nem hozhatunk létre aktív partíciót.

OUT OF ENVIRONMENT SPACE
- nincs elég hely a program-környezetben.

DOS. A rendszer nem tudta végrehajtani a SET parancsot, mert a program-környezetben már nem volt elég hely.

OUT OF ENVIRONMENT SPACE
(KEIN UMGEBUNSBEREICH MEHR VORHANDEN)
- Nincs több környezet terület

DOS. A DOS nem fogadta el a SET parancsot, mivel azt a területet nem tudta kibővíteni, Ez akkor fordulhat elő, ha egy rezidens program betöltése után újabb információkat akarunk a környezethez hozzáfűzni.(Rezidens pl. egy egyszerű billentyűzet rutin, PRINT, MODE vagy GRAPHICS parancs.)

OUT OF SPACE ON LIST FILE
(KEIN PLATZ MEHR IN LISTDATEI)
- A lista-file-ban nincs több hely

LINK. Ez a hiba akkor lép fel, ha a lemezen nincs több hely a lista file számára.

OUT OF SPACE ON RUN FILE
(KEIN PLATZ MEHR IN AUSFÜHRUNGSDATEI)
- Nincs több hely a végrehajtási file-ban

LINK. Ez a hiba akkor lép fel, ha a lemezen nincs elég hely a végrehajtási file (.EXE) számára.

OUT OF SPACE ON SCRATCH FILE

LINK. Az a lemez, amelyiken a programszerkesztő a VM.TMP (scratch) file-t letre akarja hozni, tele van.

OUT OF SPACE ON VM.TMP
(KEIN PLATZ MEHR IN VM.TMP)
- A VM.TMP-ben nincs több hely

LINK. A lemezen nincs több hely a VM.TMP file bővítésére.

P...

PARAMETERS NOT COMPATIBLE
(PARAMETER NICHT VERTRÄGLICH)

- A paraméter nem használható ebben az alakban.

Az üzenetet több parancs is küldheti. A megadott paraméterek kombinációja nem megfelelő.

PARAMETER NOT COMPATIBLE WITH FIXED DISK

- a paraméter nem használható fix lemezre.

FORMAT. A parancs paraméterben /1-et vagy /8-at adunk meg, miközben a fix lemezt formáztuk. Ezek a paraméterek fix lemezre nem érvényesek.

PARITY ERROR OR NONEXISTENT MEMORY ERROR DETECTED

DEBUG. A program a memória írása/olvasása közben paritáshibát észlelt. Ennek oka lehet az is, hogy nemlétező memóriacímet használtunk. Ha a hiba ismétlődik, jelentsük a gép szállítójának is.

PARTITION SELECTED (#) IS NOT BOOTABLE, ACTIVE PARTITION NOT CHANGED

- a kiválasztott partíció nem tölthető, az aktív partíció nem változott.

FDISK. Aktív partíció csak tölthető lehet. Ha nem ilyen partícióról van szó, azt nem tudjuk aktív partíciónak kijelölni.

PARTITON 1 IS ALREADY ACTIVE

- Az 1 partíció már aktív

FDISK. Tájékoztató üzenet: a fix lemezen egyetlen partíció van, s az már aktív.

PARTITION xx MADE ACTIVE

- Az xx partíció aktív

FDISK. Tájékoztató üzenet: a parancs az xx partíciót aktivizálta. A rendszer újraindításkor innen töltődik be.

PATH NOT FOUND

- Az út nem létezik

Valamelyik parancsban olyan utat adtunk meg, amelyik nem létezik az adott lemezegységen.

PATH TOO LONG

- Az út túl hosszú

Valamelyik parancsban 63 karakternél hosszabb utat adtunk meg.

PATHNAME TOO LONG
- Az útnév túl hosszú

PRINT

Valamelyik nyomtatandó file-hoz tartozó elérési út több mint 63 karakterből áll.

PLEASE REPLACE ORIGINAL DISKETTE IN DRIVE A: AND PRESS <ENTER>
- Helyezze vissza az eredeti lemezt az A: egységbe, majd nyomja meg az <ENTER> billentyűt

LINK. Az üzenetet a pogram az .EXE file írása előtt adja ki, feltéve, hogy a /P kulcsot megadtuk. Helyezzük be a lista file-t tartalmazó lemezt, és nyomjuk le az <ENTER> billentyűt.

PRESS ANY KEY TO BEGIN ADDING FILES
- A file-ok hozzáadásához nyomjon meg egy billentyűt.

REPLACE. Az /A /N paramétereket egyszerre adtuk meg. Ha kész vagyunk, akkor nyomjunk meg egy tetszőleges billentyűt.

PRESS ANY KEY TO BEGIN COPYING FILE(S)
- A másolás megkezdéséhez nyomjon meg egy billentyűt

XCOPY. A másolás megkezdése előtt a program megáll, hogy legyen idő a lemezek cseréjére. TetszÖleges billentyű lenyomására a másolás megkezdődik.

PRESS ANY KEY TO BEGIN FORMATTING d:
(EINE TASTE BETÄTIGEN - START FORMATIEREN d:)
- Ha lenyom egy billentyűt, az egységen elkezdődik a formázás

FORMAT. A fix lemezt (d: meghajtón) kell formázni. A formázás során valamennyi adat elvész a lemezről. A <CTRL+<BREAK> lenyomásával nem indul, más billentyű lenyomása esetén elkezdődik a formázás.

PRESS ANY KEY TO BEGIN RECOVERY OF THE FILE(S) ON DRIVE x
- a file-ok visszaállításához nyomjon meg egy billentyűt.

RECOVER. A megfelelő egységben helyezzük be azt a lemezt, amelyen a file-okat vissza akarjuk állítani, majd nyomjunk meg egy tetszőleges billentyűt.

PRESS ANY KEY TO BEGIN REPLACING FILE(S)
- A file-ok helyettesítéséhez nyomjon meg egy billentyűt

REPLACE. A parancs végrehajtása megállt, hogy ki lehessen cserélni a lemezt. Tetszőleges billentyű megnyomására folytatja a parancs végrehajtását.

PREVIOUSLY PREPARED CODE PAGE REPLACED
- az előzőleg előkészített kód-lapot kicseréltem.

MODE. A kód-lapok tárolására szolgáló hely már megtelt, ezért a PREPARE alparancs valamelyik előzőleg előkészített kód-lapot lecserélte az újra. A MODE STATUS segítségével megtudhatjuk, hogy mely kód-lapok maradtak érvényesek.

PRIMARY DOS PARTITION ALREADY EXISTS
- elsődleges DOS partíció már létezik.

FDISK. Újabb elsődleges DOS partíciót akartunk a fix lemezen létrehozni, holott csak egy lehet.

PRIMARY DOS PARTITION CREATED
- elsődleges DOS partíció létrehozva.

FDISK. A program a fix lemezen létrehozta az elsődleges DOS partíciót.

PRIMARY DOS PARTITION DELETED
- elsődleges DOS partíció törölve.

FDISK. A program törölte az elsődleges DOS partíciót.

PRINT QUEUE IS EMPTY
(WARTESCHLANGE IST LEER)
- A PRINT sor üres

PRINT. A PRINT nem dolgoz fel pillanatnyilag file-okat.

PRINT QUEUE IS FULL
(WARTESCHLANGE IST VOLL)
- A PRINT sor megtelt

PRINT. A várakozási sorba több mint 10 file-t akartunk besorolni. A várakozási sorba egy újabb file csak akkor léptethető be, ha egy nyomtatás befejeződött., vagy a sorból file-t töröltünk.

PRINTER ERROR
(DRUCKERFEHLER)
- Nyomtató hiba

MODE. A MODE parancs nem tudta a nyomtatási módot beállítani. Okai: I/O hiba; a papír kifogyott (nyomtató kikapcsolva); idő túllépés (nincs készenlétben).

PRINTER LINES PER INCH SET
(SETZEN DER DRUCKERZEILEN PRO ZOLL)
- A nyomtatott sorok száma inch-enként beállítva

MODE. Megkíséreltük a sortávolságot 6 vagy 8 inch-re állítani. Ha ez sikertelen volt, akkor ezt egy hibaüzenet követi a képernyőn.

PROBABLE NON-DOS DISK, CONTINUE (Y/N)?
(WAHRSCHEINLICH KEINE-DOS-DISKETTE, FORTFAHREN (J/N)?)
- Valószínűleg nem DOS lemez. Folytassuk (I/N)?

CHKDSK. A file-leíró táblázat (FAT) azonosító byte-jai érvénytelen információt tartalmaznak. A lemezt vagy nem DOS alatt formázták, vagy súlyosan megsérült. Ha a /F paramétert nem használtuk és Y-vel, ill J-vel felelünk, akkor a CHKDSK egy lehetséges hibaelhárítási megoldást ad anélkül, hogy a lemezt valójában változtatná.

PROCESSING CANNOT CONTINUE
(VERARBEITUNG KANN NICHT FORTGESETZT WERDEN)
- A feldolgozás nem folytatható

CHKDSK. Ezt egy további üzenet is követi, amely megadja, hogy a CHKDSK miért nem folytatható. Ennek általában kevés memória az oka.

PROGRAM SIZE EXCEEDS CAPACITY OF LINK
(PROGRAMMGRÖBE ÜBERSCHREITET KAPAZITÄT VON LINK)
- A program mérete túllépi a LINK címtartományát

LINK. A modul, amit a LINK szerkeszt, túl nagy.

PROGRAM TERMINATED NORMALLY
- A program rendben lefutott

DEBUG. Tájékoztató üzenet. A G(o), T(race) vagy P(roceed) paranccsal elindított program INT 20, INT 27, INT 21, vagy egyéb támogatott módon állt meg.

PROGRAM TOO BIG TO FIT IN MEMORY
(PROGRAMM FÜR DEN SPEICHER ZU GROB)
- A program nem fér a memóriába

DOS. A külső (external) típusokat tartalmazó file-ok nem tölthetők be, mivel méretük nagyobb, mint a rendelkezésre álló memória. Ebben az esetben a CONFIG.SYS file-ban a BUFFERS=xx értéket csökkenteni kell, majd a rendszert újra kell indítani. Ezután a parancsot újra ki lehet adni. Ha ismét ezt az üzenetet kapjuk, akkor a rendszer nem rendelkezik a parancs végrehajtásához szükséges memóriával.

R...

READ ERROR, COUNTRY.SYS
READ ERROR, KEYBOARD.SYS
- a COUNTRY.SYS(KEYBOARD.SYS) file olvasása közben hiba.

SELECT. A jelzett file olvasása közben hiba történt. Feltehetően a használt lemez tönkrement. Próbálkozzunk a biztonsági másolattal.

READ ERROR IN: x:\xxxx\xxxx
- Olvasási hiba az x:\xxxx\xxxx file-on

EDLIN. Az x:\xxxx\xxxx file olvasása közben hiba történt. A file valószínűleg fizikai hibás.

REINSERT DISKETTE FOR DRIVE D AND STRIKE ENTER WHEN READY
- Helyezze vissza a lemezt az egységbe, majd nyomjon meg egy billentyűt

FORMAT. A /S paramétert adtuk meg, de a parancs nem tudta a rendszerfile-okat egyszerre beolvasni a memóriába, mert nem volt elég hely. Ezért újra kéri a rendszerlemezt. Helyezzük be, majd nyomjuk meg az < ENTER > billentyűt.

REPLACE < d:út\file név > (Y/N)?
- Helyettesítsem a < d:út\file név > file-t (I/N)?

REPLACE. A /P paramétert adtuk meg, s a parancs visszakérdez, hogy mely file-okat akarjuk helyettesíteni. Ha N-nel válaszolunk, akkor a file-t a parancs nem helyettesíti, ha Y-nal, akkor igen.

REPLACING < d:út\file név >
- A < d:út\file név > file-t helyettesíti

REPLACE. Tájékoztató üzenet, hogy a jelzett file helyettesítése megkezdődött.

REQUESTED LOGICAL DRIVE SIZE EXCEEDS THE MAXIMUM AVAILABLE SPACE
- a kívánt logikai egység méret meghaladja a maximális rendelkezésre álló méretet.

FDISK. Az általunk megadott logikai egység mérete túl nagy. A program válaszként mindig a használható legnagyobb értéket kínálja.

REQUESTED PARTITION SIZE EXCEEDS THE MAXIMUM AVAILABLE SPACE
- a kívánt partíció-méret meghaladja a maximálisan rendelkezésre állót.

FDISK. Az általunk megadott partíció-méret túl nagy. A program válaszként mindig a használható legnagyobb értéket kínálja.

REQUESTED STACK SIZE EXCEEDS 64K
(ANGEFORDETE STAPELGRÖßE ÜBERSCHREITET 64K)
- Az igényelt verem (STACK) mérete túllépi a 64 K-t.

LINK. Ha a STACK SIZE= üzenet jelenik meg, akkor 64 K-nál kisebb értéket kell megadni.

RELOCATION TABLE OWERFLOW
- A relokációs tábla túlcsordult

LINK. A szerkesztett program több mint 13000 távoli (LONG) attributumú vezérlés-átadást és mutatót tartalmaz, ezért a LINK már nem képes ezek tárolására. Az ilyen hivatkozások számát csökkenteni kell.

RESIDENT PART OF PRINT INSTALLED
(RESIDENTER TEIL VON PRINT INSTALLIERT)
- A PRINT rezidens része installálva van

PRINT. Amikor a PRINT-et először használjuk, akkor jelenik meg ez az üzenet. Egy program a memóriába került, amely a következő PRINT parancsokat feldolgozza. Ezzel a szabad memória 3200 byte-tal csökkent.

RESIDENT PORTION OF MODE LOADED
(RESIDENTER TEIL VON MODE GELADEN)
- A MODE rezidens része be van töltve

MODE. Amikor a MODE-t először hívjuk be nem kijelzési funkcióra, akkor a MODE egy részét is rezidenssé kell tenni.

RESIDENT PORTION OF NLSFUNC LOADED
- az NLSFUNC rezidens része betöltve.

NLSFUNC. Információs üzenet: a program rezidens része a tárba töltődött.

RESTORE FILE SEQUENCE ERROR
- Visszatöltési file szám hiba

RESTORE. A file-t a parancs nem tudta visszatölteni, mert a mentett lemezeket nem megfelelő sorrendben adtuk.

***** RESTORING FILES FROM DRIVE Y *****
- Az Y egységről az X egységre a visszatöltés megkezdődött.

RESTORE. Tájékoztató üzenet, mely jelzi, hogy az y egységről az x egységre megkezdődött a file-ok visszatöltése.

S...

SAME DRIVE SPECIFIED MORE THAN ONCE
- ugyanarra az egységre többször hivatkoztunk.

FASTOPEN. A parancsban egy-egy lemezegység betűjele csak egyszer szerepelhet.

SECTOR SIZE ADJUSTED
- Szektor mérete módosítva

VDISK. A konfigurációs parancs módosította az általunk megadott szektorméretet. Nem hibaüzenet.

SECTOR SIZE TOO LARGE IN FILE < FILENAME >
(SEKTORGRÖBE IN DATEI < DATEINAME > ZU GROB)
- A szektor mérete túl nagy a < filenév > file-ban

START. Az operációs rendszer töltésekor a < filenév > periféria kezelőben a DOS számára definiáltnál nagyobb szektorméretet adtunk meg. A paramétert módosítani kell, majd újraindítani a rendszert.

SEGMENT LIMIT TOO HIGH, EXCEEDS 1024
- Túl nagy szegmensszámot adtunk meg, a maximum 1024

LINK. A /x paraméterben megadott szegmensszám nagyobb a megengedettnél. Módosítsuk, és szerkesszük újra a programot.

SEGMENT LIMIT TOO HIGH
- Túl nagy szegmensszámot adtunk meg

LINK. A /x paraméterben (kezdő értéke 256) megadott számú szegmens részére a LINK nem tud elegendő helyet foglalni a memória táblázataiban. Csökkentsük a /x paraméter értékét, vagy szabadítsunk fel további memóriaterületet (pl. a VDISK-et).

SEGMENT SIZE EXCEEDS 64K
(SEGMENTGRÖBE ÜBERSCHREITET 64K)
- A szegmens mérete túllépi a 64 K-t

LINK. Szerkesztés során egy szegmens mérete túllépi az 64 K-t. A legnagyobb címtartomány viszont csak 64 Kbyte.

SHARE ALREADY INSTALLED
- A file megosztási környezetet már installáltuk.

SHARE. A file megosztási környezetet csak egyszer lehet betölteni, újra megkíséreltük. Nem hibaüzenet.

SHARING VIOLATION
- File megosztási hiba

XCOPY. A cél vagy forrás file-ok valamelyikét egy másik processzo zárolta, ezért az XCOPY nem tudja írni vagy olvasni. Várjunk, míg a file felszabadul, s adjuk újra ki a parancsot.

SOURCE AND TARGET DRIVES ARE THE SAME
- A forrás és a cél lemez ugyanaz

BACKUP vagy RESTORE. A forrás és a cél lemezegységet azonosnak adtuk meg. Adjunk meg különböző egységeket és adjuk ki újra a parancsot.

SOURCE DOES NOT CONTAIN BACKUP FILES
- A forrás lemezen nincsenek mentett file-ok

RESTORE. A parancsban megjelölt forrás lemezen nincsenek mentett file-ok, ezért a parancsot nem lehet végrehajtani.

SOURCE / PATH REQUIRED
- A forrás-hoz vezető utat meg kell adni

REPLACE. A parancsban nem szerepel egyetlen forrás út sem, holott legalább egyet meg kell adni.

SPECIFIED COMMAND SEARCH DIRECTORY BAD
- A parancshoz vezető út hibás

Akármelyik parancs küldheti. A parancsot megelőző lemez és útleírás rossz. Kijavítása után ismételjük meg a parancsot.

SPECIFIED DRIVE NOT EXIST, OR IS NON-REMOVABLE
- A megadott lemezegység nem létezik, vagy fix

DISKCOPY vagy DISKCOMP. A parancsban megadott lemezegység nincs a rendszerben vagy fix lemez. Javítsuk ki a lemezegység azonosítóját és adjuk ki újra a parancsot.

SOURCE DISKETTE BAD OR INCOMPATIBLE
- A forrás lemez rossz vagy nem megfelelő formájú

DISKCOPY. A lemez olvasása közben hiba történt. Ennek oka lehet, hogy bizonyos szektorok fizikailag tönkrementek, de az is, hogy egy 360 Kbyte-os lemezegységbe egy nagy kapacitású lemezt helyeztünk.

STACK SIZE EXCEEDS 65535 BYTES
(STAPELGRÖBE ÜBERSCHREITET 65535 BYTES)
- A verem mérete túllépi a 65535 byte-ot

LINK. A megadott méretnek kisebbnek kell lennie mint 65535 byte.

SYMBOL DEFINED MORE THAN ONCE
(SYMBOL MEHRFACH DEFINIERT)
- Többszörösen definiált szimbólum

A LINK olyan modulokat talált, amelyek ugyanazt a szimbólumot definiálják.

SYMBOL TABLE OVERFLOW. SYMBOL TABLE CAPACITY EXCEEDED
(KAPAZITÄT DER SYMBOLTABELLE ÜBERSCHRITTEN)
- A szimbólum tábla méretét túlléptük

LINK. Túl sok és túl hosszú nevet adtunk meg. Méretük meghaladja a 256 Kbyte-ot. Rövidebb és kevesebb szimbólumot kell használni.

SYNTAX ERROR
(SYNTAXFEHLER)
- Szintaktikus hiba

DOS. A parancs formája helytelen. Ellenőrizni kell a parancs formáját.

SYSTEM FILES RESTORED. THE TARGET DISK MAY NOT BE BOOTABLE
- Rendszer file-ok visszatöltve. Lehet, hogy a lemez nem betölthető

RESTORE. A DOS rendszerfile-okat (IBMBIO.COM, IBMDOS.COM) visszatöltöttük. Előfordulhat, hogy a lemez nem lesz betölthető, mert korábbi DOS verziót írtunk rá.

SYSTEM TRANSFERED
- A rendszer átvitele megtörtént

FORMAT. Az üzenet a /S paraméter használatakor jelenik meg, jelezve, hogy a DOS rendszerfile-ok is rákerültek a lemezre.

SYSTEM WILL NOW RESTART. INSERT DOS DISKETTE IN DRIVE A:, PRESS ANY KEY WHEN READY
- A rendszer újraindul. Helyezze be a DOS lemezt az A: meghajtóba, majd nyomjon meg egy billentyűt

FDISK. Tájékoztató üzenet: az FDISK használata után újra kell indítani a rendszert, hogy az újjáírt partíciós táblát használja. Helyezzük be a DOS lemezt az A: egységbe, és nyomjunk meg egy tetszőleges billentyűt.

T...

TARGET CANNOT BE USED FOR BACKUP
- A cél lemez nem használható mentésre

BACKUP. A cél lemez használhatatlan. Ha cserélhető, próbáljuk meg egy másik lemezzel. Ha fix, akkor indítsuk újra a rendszert és adjuk ki újra a parancsot.

TARGET DISKETTE MAY BE UNUSABLE
(ZIELDISKETTE MÖGLICHERWEISE UNBRAUCHBAR)
- Lehetséges, hogy a cél lemez nem használható

DISKCOPY. Ez az üzenet egy olyan hibaüzenet után áll, amely egy nem ismételten fellépő írási vagy olvasási hibára utal. A céllemezt I/O hiba miatt már nem lehet majd másolni.

TARGET DISKETTE UNUSABLE
- A cél lemez használhatatlan

DISKCOPY. A másolás során a cél lemezen hibát talált a rendszer. Egy másik lemezről ismételjük meg a másolást.

TARGET DISKETTE WRITE PROTECTED. CORRECT, THEN STRIKE ANY KEY
(ZIELDISKETTE IST SCHREIBGESCHÜTZT, BERICHTIGEN UND EINE TASTE BETÄTIGEN)
- A cél lemez írásvédett. Javítsa ki, majd nyomjon le egy billentyűt

DISKCOPY. Írásvédett lemezre akartunk másolni.

TARGET IS FULL
- A cél lemez megtelt

RESTORE. Mentés közben a lemez megtelt. Próbáljuk meg a mentést egy másik lemezre, amin több szabad hely van.

TERMINATE BATCH JOB (Y/N)?
(STAPELJOB BEENDEN (J/N)?
- A kötegelt (BATCH) job befejeződjön (I/N)?

DOS. Ez az üzenet akkor jelenik meg, ha a <CTRL-BREAK> billentyűt nyomjuk le, miközben a DOS parancs file-t dolgoz fel. Ha a feldolgozást be akarjuk fejezni, akkor a Y-t, ill. a J-t kell lenyomni. Ha N-t adunk meg, akkor csak az a parancs fejeződik be, amely éppen feldolgozás alatt állt. A feldolgozás a következő paranccsal folytatódik.

TERMINATE BY USER
- A felhasználó megszakította a programot

LINK. A <CTRL-BREAK> billentyűvel megállítottuk a programot, amit már csak újraindítani lehet. Tájékoztató üzenet.

THE CURRENT ACTIVE KEYBOARD TABLE IS xx
WITH CODE PAGE: yyy
THE CURRENT ACTIVE CON CODE PAGE IS: zzz
- a jelenlegi aktív billentyűzet: xx
- a kód-lap: yyy
- a CON eszköz kód-lapja: zzz

KEYB. A parancs lekérdező funkciójának használatakor kapjuk a fentieket válaszul. Információs üzenet.

THE CURRENT ACTIVE PARTITON IS X
(DIE MOMENTANE AKTIVE PARTITION IST X)
- A pillanatnyilag aktív partició az x

FDISK. Az "aktív partíció változtatása" lehetőség kijelzi a fix lemez pillanatnyilag aktív partícióját.

THE LAST FILE WAS NOT RESTORED
(DIE LETZTE DATEI WURDE NICHT ZURÜCKGESPEICHERT)
- Az utolsó file visszatöltése nem történt meg

RESTORE. A visszatöltés befejeződött, mielőtt a megadott file-okat teljezen visszatöltötte volna, vagy kevés volt a hely a fix lemezen. A RESTORE a részben visszatöltött file-t törölte.

THE ONLY BOOTABLE PARTITION ON DRIVE 1 IS ALREADY MARKED ACTIVE
- az első fix lemez egyetlen tölthető partíciója már aktív.

FDISK. Információs üzenet.

THERE WAS/WERE xxx ERRORS DETECTED
(ES WURDE(N) ZAHL FEHLER ENTDECKT)
- A szerkesztés hibaszintje

LINK. A szerkesztés végén jelenik meg ez a tájékoztató üzenet. A szerkesztés sikerességéről, illetve sikertelenségéről kapunk információt.

TOO MANY BLOCKS DEVICES
- Túl sok blokkos periféria kezelő

CONFIG.SYS. A rendszer maximum 26 blokkos perifériakezelőt tartalmazhat. A konfigurációs file-ban ennél többet kíséreltünk meg implementálni. Csökkentsük ezek számát és indítsuk újra a rendszert.

TOO MANY DRIVE ENTRIES
- túl sok lemezegység megadása

FASTOPEN. A parancsban túl sok egységet soroltunk fel. Csökkentsük ezek számát, majd ismételjük meg a parancsot.

TOO MANY EXTERNAL SYMBOLS, LIMIT 510 PER MODULE. TOO MANY EXTERNAL SYMBOLS IN ONE MODULE
(ZU VIELE EXTERNE SYMBOLE IN EINEM MODUL)
- Túl sok külső hivatkozás egy modulban

LINK. Egy modulban legfeljebb 510 külső hivatkozás (external reference) lehet.

TOO MANY FILES OPEN
- Túl sok nyitott file

EDLIN. A parancs megkísérelte a megadott file-t megnyitni, de nem tudta. Növeljük meg a konfigurációs file-ban a FILES = értéket, és indítsuk újra a rendszert.

TOO MANY GROUP-, SEGMENT-, AND CLASS-NAMES, LIMIT 254 PER MODULE
- Túl sok csoport-, szegmens-, és osztály név, a maximum 254 modulonként

TOO MANY GROUPS
(ZU VIELE GRUPPEN)
- Túl sok csoport

LINK. A csoportok száma a DGROUP-pal együtt legfeljebb 10 lehet.

TOO MANY GRPDEFS, LIMIT 8 PER MODULE
- Túl sok GRPDEF, modulonként 8 lehet.

LINK. A szerkesztő valamelyik modulban több mint 8 GRPDEF utasítást talált. A számukat csökkenteni kell.

TOO MANY LIBRARIES SPECIFIED
TOO MANY LIBRARIES, LIMIT IS 16
(ZU VIELE BIBLIOTHEKEN ANGEGEBEN)
- Túl sok könyvtárat adtunk meg

LINK. Több mint 8 könyvtárat adtunk meg.

TOO MANY NAME ENTRIES
- túl sok név megadása

FASTOPEN. Összesen 999 file- és út-név adható meg. A feleslegeseket hagyjuk el és ismételjük meg a parancsokat.

TOO MANY OPEN FILES
- túl sok nyitott file

XCOPY. A rendszert úgy konfiguráltuk, hogy a parancs nem tud kellő file-kezelőt használni. A CONFIG.SYS file-ban a FILES paraméterét legalább 20-nak válasszuk meg és indítsuk újra a rendszert.

TOO MANY OVERLAYS
(ZU VIELE ÜBERLAGERUNGEN)
- Túl sok az overlay

LINK Az overlay mérete legfeljebb 65535 lehet.

TOO MANY PUBLIC SYMBOLS
(ZU VIELE ALLGEMEINE SYMBOLE)
- Túl sok közös szimbólum

LINK. A közös szimbólumok legfeljebb 1024 byte-ot foglalhatnak el.

TOO MANY SEGMENTS, LIMIT 255 PER MODULE
- Túl sok szegmens, a maximum 255 lehet modulonként

LINK. Az objekt program több mint 255 szegmenst tartalmaz, ami nem megengedett. Csökkentsük a szegmensek számát, és futtassuk újra a LINK-et.

TOO MANY SEGMENTS, USE /X:N (256 < N < 1025)
- Túl sok szegmens, használja a /X:N (256 < N < 11025) kulcsot

LINK. A lehetséges-nél több szegmens szerepel a szerkesztő programban. Az /X paraméter megadásával ill. a szegmensek számának csökkentésével szerkesszük újra a programot.

TOO MANY TYPEDEFS, LIMIT 255 PER MODULE
- Túl sok TYPEDEF, a maximum 255 lehet

LINK. A TYPEDEF rekordokat a programszerkesztő a közös (COMMON) változók leírására használja, egy modulban legfeljebb 255 lehet, s ezt a határt túlléptük.

TOP LEVEL PROCESS ABORTED, CANNOT CONTINUE
- A legfelső szintű processz abortált, nem lehet folytatni

A rendszer betöltése közben a COMMON.COM vagy egy másik parancs lemezhibát jelzett, s az ABORT-ot választottuk. Másik lemezzel kell a rendszert indítani.

TOTAL DISC SPACE IS XXXX CYLINDERS
(GESAMTER PLATTENBEREICH BETRÄGT XXXX ZYLINDER)
- A teljes lemezterület mérete xxxx cilinder

FDISK. A teljes fix lemezterületet kiírja

TRACK 0 BAD-DISC UNUSABLE
(SPUR 0 FEHLERHAFT - DISKETTE UNBRAUCHBAR)
- A 0. sáv a lemezen nem használható

FORMAT. A lemez 0. sávján a következők helyezkednek el: boot-rekord, tartalomjegyzék rekord, file-leíró táblázat (FAT). Valamelyik szektor hibás, a lemez használhatatlan.

TRANSFER SIZE ADJUSTED
- Az átvitel mérete módosítva

VDISK. Az egyszerre átvihető rekordok száma nem esett az 1-8 intervallumba, ezért a VDISK 8-ra állította be.

TREE PAST THIS POINT NOT PROCESSED
(VERZEICHNISSTRUKTUR AB DIESEM PUNKT NICHT VERARBEITET)
- A tartalomjegyzék struktúrát e ponttól nem tudja feldolgozni

CHKDSK. A CHKDSK az éppen átvizsgált elérési út feldolgozását nem tudja folytatni, mivel a 0. sáv hibás.

U...

UNABLE TO COPY KEYBOARD ROUTINE
- A billentyűzet rutint nem lehet másolni

A KEYBxx.COM file megnyitása és olvasása közben hiba történt. Ha ezt a file tárolása okozta, később próbálkozzunk újra.

UNABLE TO CREATE DIRECTORY
(INHALTSVERZEICHNIS KANN NICHT ERSTELLT WERDEN)
- Nem lehet a tartalomjegyzéket létrehozni

MKDIR. Az előállítandó altartalomjegyzék már létezik; vagy az elérési út, amelyet a felhasználó megadott, nem található; vagy a főtartalomjegyzék már megtelt, és nem lehet újabb bejegyzést hozzáfűzni.

UNABLE TO CREATE KEYB TABLE IN RESIDENT MEMORY
- a KEYB a táblázatait nem tudja elhelyezni a rendelkezésére álló helyen.

KEYB. A parancs rezidens részét már előzőleg a memóriába töltöttük. A most kiadott parancs táblázatai nem férnek el az előzőleg erre a célra foglalt helyen. Indítsuk újra a DOS-t és a KEYB-t a megfelelő konfigurációval töltsük a memóriába.

UNABLE TO WRITE BOOT
(EINLEITENDER LADASATZ KANN NICHT GESCHRIEBEN WERDEN)
- A boot írása nem lehetséges

FORMAT. A lemez első sávja vagy a DOS partíció hibás. A kezdeti betöltőt (boot) nem lehetett erre a sávra felírni. A lemez nem használható DOS rendszerlemezként.

UNABLE TO SHIFT SCREEN LEFT
- A képernyőt nem lehet balra tolni

MODE. A képernyő a megjelölt értékkel már nem tolható el balra.

UNABLE TO SHIFT SCREEN RIGHT
- A képernyőt nem lehet jobbra tolni.

MODE. A képernyő a megjelölt értékkel már nem tolható el jobbra.

UNABLE TO WRITE BOOT
- a boot rekordot nem lehet felírni.

FORMAT. A lemez első sávjának első szektora rossz. Dobjuk ki.

UNEXPECTED END-OF-FILE ON LIBRARY
(UNERWARTETES DATEIENDE IN DER BIBLIOTHEK)
- Nem várt EOF a könyvtárban

LINK. Ez általában a könyvtári állományban levő hibára vezethető vissza.

UNEXPECTED END-OF-FILE ON SCRATCH FILE
UNEXPECTED END OF FILE ON VM.TMP
(UNERWARTETES DATEIENDE IN VM.TMP)
- Nem várt EOF a VM.TMP-ben

LINK. Például a VM.TMP file-t tartalmazó lemezt kivettük.

UNRECOGNIZED COMMAND IN CONFIG.SYS
(NICHT ERKANNTER BEFEHL IN CONFIG.SYS)
- Nem felismerhető parancs a CONFIG.SYS-ben

START. Érvénytelen parancs található a CONFIG.SYS file-ban. A hibás parancsot ki kell javítani, majd a rendszert újra kell indítani.

UNRECOGNIZED SWITCH ERROR: XXXXXX
- Az xxxxxx kapcsoló rossz

LINK. A / jel után egy nem azonosítható xxxxx kulcsot használtunk.

UNRECOVERABLE ERROR ON DIRECTORY
- A főtartalomjegyzék rossz

CHKDSK. A főtartalomjegyzék ellenőrzése közben a parancs hibát talált.

UNRECOVERABLE FILE SHARING ERROR
- Felismerhetetlen file megosztási hiba

Valamelyik parancs azonosíthatatlan file megosztási hibát észlelt.

UNRECOVERABLE FORMAT ERROR ON TARGET. DISKETTE UNUSABLE
(NICHT WIEDERHERSTELLBARER FORMATFEHLER AUF ZIELDISKETTE / ZIELDISKETTE UNBRAUCHBAR)
- Nem javítható formátum hiba a cél lemezen. A cél lemez használhatatlan.

DISKCOPY. Nem javítható hiba a cél lemez formázásánál. A lemez hibás adatokat tartalmaz.

UNRECOVERABLE READ ERROR ON DRIVE X TRACK XX SIDE X
(NICHT WIEDERHERSTELLBARER LESEFEHLER IN LAUFWER X SPUR XX, SEITE X)
- Nem javítható olvasási hiba az X meghajtón az XX sávon X oldalon

DISKCOMP. Az adatok olvasása a megadott meghajtóban lévő lemezen négy kísérlet során sem sikerült. A hiba helyét az üzenet megadja.

UNRECOVERABLE READ ERROR ON SOURCE TRACK XX, SIDE X
(NICHT WIEDERHERSTELLBARER LESEFEHLER AUF QUELLENDISKETTE SPUR XX, SEITE X)
- Nem javítható olvasási hiba a forráslemez x oldalán xx sávon

DISKCOPY. A forráslemezről négy sikertelen olvasás történt. A DISKCOPY folytatja az olvasást, de a másolat hiányos adatokat tartalmazhat.

UNRECOVERABLE VERIFY ERROR ON TARGET TRACK XX, SIDE X
(NICHT WIEDERHERSTELLBARER PRÜFFEHLER AUF ZIELDISKETTE SPUR XX, SEITE X)
- Nem javítható ellenőrzési hiba a cél lemez x oldalán, xx sávján

DISKCOPY. Négy sikertelen írás-ellenőrzés volt a cél lemezen, a DISKCOPY folytatja a másolást, de a másolat hiányos adatokat tartalmazhat.

UNRECOVERABLE WRITE ERROR ON TARGET TRACK XX, SIDE X
(NICHT WIEDERHERSTELLBARER SCHREIBFEHLER AUF ZIELDISKETTE SPUR XX, SEITE X)
- Nem javítható írás hiba a cél lemez x oldalán, xx sávján

DISKCOPY. Négy sikertelen írás volt a cél lemezre. A DISKCOPY folytatja a másolát, de a másolat hiányos adatokat tartalmazhat.

UNRESOLVED EXTERNALS: LIST
(UNBEKANNTE EXTERNE SYMBOLE: LISTE)
- Feloldhatatlan külső hivatkozás: lista

LINK. A kilistázott külső hivatkozások (external reference) definiálatlanok a modulokban. Az így előállított load-modult nem szabad végrehajtani.

V...

VDISK NOT INSTALLED - INSUFFICIENT MEMORY
- A memória lemez - memóriahiány miatt - nincs installálva

VDISK. A memória mérete nem megfelelő. Ennek oka lehet, hogy a paraméterek korrigálása után is 64 Kbyte-nál kisebb használható memória van, de lehet az is, hogy a kiterjesztett memória mérete nem megfelelő.

VDISK VERSION v.vv VIRTUAL DISK X
- VDISK verzió v.vv Memória lemez: x

VDISK. A memória lemez sikeres installálásáról tájékoztató üzenet. A memória lemezre az x betűvel hivatkohatunk.

VERIFY IS ON/OFF
- Ellenőrzés be/ki kapcsolva

VERIFY. Tájékoztató üzenet. ON = a rendszer ellenőrzi a lemezre történő írást, OFF = nem.

VM.TMP IS AN ILLEGAL FILENAME AND HAS BEEN IGNORED
(VM.TMP IST EIN UNZULASSIGER DATEINAME UND WURDE IGNORIERT)
- VM.TMP illegális file-azonosító, ezért figyelmen kívül marad

LINK. A VM.TMP-t tárgy (object) modul azonosítóként nem lehet megadni.

VOLUME LABEL (11 CHARACTERS, ENTER FOR NONE)?
(VOLUME-NAME (11 ZEICHEN, KEINEE NAME->EINGABE)?)
- A kötetazonosító (11 karakter, ha nem kell, ENTER)?

FORMAT. Ez az üzenet a kötetazonosítót kéri, ami legfeljebb 11 karakter lehet. Ha a felhasználó nem akar kötetazonosítót felírni, akkor csak az ENTER billentyűt kell megnyomnia.

W...

WARNING! ALL DATA ON NON-REMOVABLE DISK DRIVE X WILL BE LOST PROCEED FORMAT (Y/N)?
- Figyelem! Az x fix lemez összes adata elvész. Megformázzam (I/N)?

FORMAT. Fix lemezegység formázása törli az összes adatot. Ha Y-nal válaszolunk, akkor a formázás megtörténik, ha N-nel, akkor nem.

WARNING! ALL DATA IN THE DOS PARTITON WILL BE DESTROYED. DO YOU WISH TO CONTINUE...? [d]
(ACHTUNG! ALLE DATEN IN DER DOS PARTITON WERDEN ZERSTÖRT. SOLL FORTGEFAHREN WERDEN...? [d])
- Figyelem! Minden adat a DOS partícióban elvész. Akarja folytatni...? [d]

FDISK. A "DOS partíció törlés" lehetőség választása esetén jelenik meg ez az üzenet. A DOS partíció valamennyi adata törlésre kerül. Ha csak az <ENTER> billentyűt nyomjuk le, akkor nem törlődnek, ha viszont az Y-t, ill. a J-t és üzenetvég-et is adunk, akkor törlődnek.

WARNING! DATA IN THE EXTENDED DOS PARTITION COULD BE DESTROYED. DO YOU WISH TO CONTINUE?
- Vigyázat: a kibővített DOS partíción levő adatok megsemmisülhetnek. Akarja folytatni?

FDISK. A kibővített DOS partíció törlésekor kapjuk a fenti üzenetet. Ha ennek ellenére törölni akarjuk a partíciót, akkor az <4> billentyű után nyomjuk meg az <ENTER>-t!

WARNING! DATA IN THE PRIMATY DOS PARTITION COULD BE DESTROYED. DO YOU WISH TO CONTINUE?
- Vigyázat: az elsődleges DOS partícióban levő adatok megsemmisülhetnek. Akarja folytatni?

FDISK. Az elsődleges DOS partíció törlésekor kapjuk a fenti üzenetet. Ha ennek ellenére törölni akarjuk a partíciót, akkor az < Y > billentyű után (Yes) nyomjuk meg az < ENTER > -t is!

WARNING! DIRECTORY FULL
xxx FILE(S) RECOVERED
(WARNUNG! INHALTSWERZEICHNIS VOLL)
- Figyelem! A tartalomjegyzék megtelt

RECOVER. Az újraelőállítandó file-ok számára nincs hely a tartalomjegyzékben. Néhány file-t egy üres lemezre kell másolni, majd az eredetiről törölni. Ezután újra végrehajtható a RECOVER.

WARNING! DISKETTE IS OUT OF SEQUENCE. REPLACE THE DISKETTE, OR CONTINUE. STRIKE ANY KEY WHEN READY
(WARNUNG! DISKETTE NICHT FOLGERICHTIG. DISKETTE AUSTAUSCHEN. WENN BEREIT, EINE TASTE BETÄTIGEN)
- Figyelem! Lemez-sorrend hiba, cserélje a lemezt, vagy nyomjon le egy tetszőleges billentyűt

RESTORE. A tartalék (backup) lemez nem a következő a sorban. A lemezt le kell cserélni. Ha a felhasználó biztos abban, hogy az átlépett (kihagyott sorszámú) lemezen nincs olyan file, amit elő kell állítani, akkor a RESTORE egy tetszőleges billentyű lenyomásával folytatható. Ez az üzenet állandóan ismétlődik, ha a felhasználó olyan lemezt akar átlépni, amelyen az újra előállítandó file egy része található.

WARNING! FILE XX IS A READ-ONLY FILE. REPLACE THE FILE (Y/N)?
(WARNUNG! DATEI XX IST EINE NUR-LESE-DATEI. DATEI AUSTAUSCHEN (J/N)?)
- Figyelem! Az xx file csak olvasható. Cserélje a file-t (I/N)?

RESTORE. A megadott file csak olvasható. Ha ki akarjuk cserélni, akkor Y-t, ill. J-t, különben N-t kell megadnunk. Az < ENTER > lenyomásával folytatható a művelet. Ezt az üzenetet csak a /P paraméter megadása esetén kapjuk.

WARNING! FILE XX WAS CHANGED AFTER IT WAS BACKED UP. REPLACE THE FILE (Y/N)?
(WARNUNG! DATEI XX WURDE NACH DER SICHERUNG GEÄNDERT. DATEI AUSTAUSCHEN (J/N)?)
- Figyelem! Az xx file a mentés óta megváltozott. Az állományt lecseréljük (I/N)?

RESTORE. A megadott file a fix lemezen más dátumot és időt tartalmaz, mint a tartalék lemezen található kimentett file. Az Y, ill. J válasz magadásával a file-t lecseréljük. A

RESTORE az < ENTER > lenyomása után folytatódik. Ezt az üzenetet a /P paraméter megadása esetén kapjuk.

WARNING! FILES IN THE TARGET DRIVE d:\ ROOT DIRECTORY WILL BE ERASED. STRIKE ANY KEY WHEN READY
- Figyelem! A d: egység főtartalomjegyzékében levő valamennyi file elvész. Ha rendben, nyomjon le egy billentyűt

BACKUP. Figyelmeztetés: a parancs a főtartalomjegyzékben levő valamennyi file-t törölni fogja. A végrehajtás akkor kezdődik, ha megnyomunk egy tetszőleges billentyűt.

WARNING! NO FILES WERE FOUND TO BACK UP
(ACHTUNG! KEINE DATEIEN ZUR SICHERUNG GEFUNDEN)
- Figyelem! A mentésre kijelölt file nem található.

BACKUP. A mentésre kijelölt file nem található a fix lemezen.

WARNING! NO FILES WERE FOUND TO RESTORE
(ACHTUNG! KEINE DATEIN ZUM ZURÜCKSPEICHERN GEFUNDEN)
- Figyelem! A visszatöltésre kijelölt file nem található

RESTORE. A lemezen nem található a file, amelyet a fix lemezre vissza akarunk tölteni.

WARNING! NO STACK SEGMENT
(ACHTUNG! KEIN STAPELSEGMENT)
- Figyelem! Nincs verem szegmens

LINK. A megadott tárgy modulok egyike sem tartalmaz hivatkozást a verem (stack) szegmensre. (Ha .COM-ot akarunk generálni a programból, ennek az üzenetnek szerkesztéskor meg kell jelennie.)

WARNING! READ ERROR ON EXE FILE
(ACHTUNG! LESEFHELER IN EXE-DATEI)
- Figyelem! Olvasási hiba az .EXE kiterjesztésű file feldolgozásakor

EXE2BIN. A file olvasása során hiba jelentkezett. Az EXE2BIN megpróbálja a továbblépést, de az eredmény file bizonyos esetekben használhatatlanná válik. Próbálja a file-t másik meghajtóról olvasni.

WARNING! TARGET IS FULL
- Figyelem! A cél lemez megtelt

BACKUP vagy RESTORE. A cél lemez megtelt. Nem lehet további file-okat feldolgozni.

WARNING! THE PARTITION MARKED ACTIVE ISNOT BOOTABLE
- Vigyázat: az aktív partíció nem tölthető.

FDISK. A program figyelmeztet: az aktív partíció nem tölthető. Ha ez így marad, akkor a fix lemezről nem tudjuk a rendszert elindítani. Vélhetően nem ez volt a szándékunk.

WRITING xxx BYTES
- xxx byte kiírása megkezdődött

DEBUG. Tájékoztató üzenet, hány byte-ot ír ki a program.

X...

X IS NOT A CHOISE. ENTER A CHOISE.
(X IST KEINE AUSWAHL. EINGABE EINER AUSWAHL)
- Az x nem választható. Egy másik lehetőséget adjon meg.

X IS NOT CHOISE. ENTER Y OR N
(X IST KEINE AUSWAHL. EINGABE VON J ODER N)
- Az x nem választható. Megadható válasz Y, ill. J, vagy N.

XXX FILES ADDED
- xxx file-t adott hozzá

REPLACE. Tájékoztató üzenet: a parancs a cél lemezhez xxx file-t adott hozzá.

XXXXXX IS NOT A VALID LIBRARY
- xxxxxx rossz könyvtár

LINK. A könyvtár file megadása rossz.

XXXXXXXXXX BYTES DISC SPACE FREED
(XXXXXXXXXX BYTES PLATTENSPEICHER FREIGEMACHT)
- xxxxxxxxxx byte lemez terület szabaddá téve

CHKDSK. Az előzőleg kijelölt terület nem került lefoglalásra. Ez most felszabadult.

xxxxxxxx CODE PAGE DRIVER CANNOT BE INITIALIZED
- az xxxxxxxx kód-lap kezelőt nem lehet inicializálni.

PRINTER.SYS vagy DISPLAY.SYS. A DEVICE konfigurációs parancsban rossz paramétereket adtunk meg. Ellenőrizzük a CONFIG.SYS file-t, majd indítsuk újra a rendszert.

xxxxxxxx DEVICE DRIVER CANNOT BE INITIALIZED
- az xxxxxxxx periféria-kezelőt nem lehet inicializálni.

DOS. A DEVICE konfigurációsparancsban az xxxxxxx kezelőre vonatkozó parancs rossz. Ellenőrizzük a CONFIG.SYS file-t, majd indítsuk újra a rendszert.

XXXX ERROR ON FILE YYYY
(XXXX FEHLER IN DATEI YYYY)
- xxxx hiba az yyyy file-ban

PRINT. Ez az üzenet a nyomtatón jelenik meg. Miközben yyyy file-ból a nyomtatás számára adatokat olvastunk, egy xxx típusú lemezhiba lépett fel. A file kiírása befejeződött.

XXX LOST CLUSTERS FOUND IN YYY CHAINS
(XXX VERLORENE BEREICHE GEFUNDEN IN YYY BEREICHEN)
- xxx elveszett csoport található yyy darab láncban

CHKDSK. Az xxx csoportok elveszett blokkjait megtalálta a CHKDSK. A CHKDSK lehetővé teszi azt, hogy ezeket tötöljük, vagy önálló file-okként előállítsuk.

XXXXXX OF YYYYYY BYTES RECOVERED
- yyyyyy byte-ból xxxxxx-et visszaállított

RECOVER. Tájékoztató, hogy összesen hány byte-nyi információt sikerült visszaállítani, s hogy mennyiből.

***** BACKING UP FILES TO DISKETTE XX *****
(*** SICHERUNG VON DATEIEN AUF DISKETTE XX***)
- *** A file-ok mentese xx lemezre történt

BACKUP. Ezt az üzenetet a kimentett file-ok listája követi az adott lemezről.

***** FILES WERE BACKED UP XX/XX/XXXX *****
(*** DATEIN WURDEN GESICHERT XX/XX/XXXX ***)
- *** A file mentése történt xx/xx/xxxx***

RESTORE. A megadott időpontban történt a mentés. (A dátum formája attól függ, hogy a CONFIG.SYS file-ban a nemzet-specifikus paramétert beállítottuk-e.)

***** RESTORING FILES FROM DISKETTE XX *****
(***ZURÜCKSPEICHERN VON DATEIEN VON DISKETTE XX ***)
- *** A file-ok visszatöltése xx lemezről történik ***

RESTORE. Ezt követi az ismét előállított (visszatöltött) file-ok listája az adott lemezről.

nnn VERSION OF GRAPHIC CHARACTER SET TABLE IS ALREADY LOADED
- a grafikus karakterkészlet nnn verziója már a memóriában van.

GRAFTABL. A paranccsal egy már a memóriában levő és még használhatókód-lapot akartunk betölteni. Információs üzenet. nnn lehetséges értékei: 437, 860, 863 és 865.

nnn VERSION OF GRAPHIC CHARACTER SET TABLE IS NOW LOADED
- a grafikus karakterkészlet nnn verziója betöltve.

GRAFTABL. Információs üzenet.

-MORE-
(-FORTSETZUNG-)
- **Folytatás**

MORE. A képernyő megtelt, de további adatok várnak kijelzésre. A következő képernyőtöltés egy tetszőleges billentyű lenyomásával kezdeményezhető.

10 MISMATCES - ENDING COMPARE
(10 VERGLEISCHFEHLER - VERGLEICHEN BEENDET)
- 10 összehasonlítási hiba - befejezve az összehasonlítás

COMP. Már 10 nem egyező byte van az összehasonlítandó file-okban. A további összehasonlításnak nincs értelme.

B. Függelék

Parancsok, alparancsok csoportosított betűrendes listája

Parancsok	Funkció
APPEND	Bejárási utak kijelölése tetszőleges kiterjesztésű file-okra.
ASSIGN	A periféria-hozzárendelések megváltoztatására használható, például floppy-meghajtó esetén.
ATTRIB	A file-ok archiválási és csak-olvasható jelzőinek ki- ill. bekapcsolása.
BACKUP	A fix lemezen levő file-ok (esetleg a teljes fix lemez tartalmának) kimentése.
BREAK	A BREAK billentyű figyelésének beállítása, állapotának lekérdezése.
CHCP	Kód-lap váltás valamennyi eszközre.
CHDIR	Az aktuális tartalomjegyzék változtatása. *Rövidített formája*: CD.
CHKDSK	A lemez állapotáról ad információt. Hiba esetén a fizikai tárolásnak megfelelően javítja a file-leíró táblázatot (FAT).
CLS	A képernyő törlése.
COMMAND	Második parancsfeldolgozó processzor elindítása.
COMP	File-ok összehasonlítása. Az összehasonlításra kerülő file-oknak azonos méretűnek kell lenni. Legfeljebb tíz eltérést jelez ki.
COPY	File-ok másolása, duplikálása, összekapcsolása. Bináris és szöveges (text) állományok is összefűzhetők.
CTTY	Alternatív konzol kijelölése.
DATE	A rendszer-dátum lekérdezése, beállítása.
DEL	File-ok törlése a tartalomjegyzékből.
DIR	A file-ok tartalomjegyzék bejegyzésének listázása. Teljes és tömörített formában is használható.
DISKCOMP	Floppy lemezek összehasonlítása (fizikai, szektoronként).
DISKCOPY	Floppy lemezek fizikai másolása.
ERASE	File-ok törlése a tartalomjegyzékből. *Rövidített formája*: DEL.
EXE2BIN	A végrehajtható .EXE formából .COM formájú programot készít.
EXIT	Kilépés a második COMMAND-ból.
FASTOPEN	File-ok megtalálásának lerövidítése.
FDISK	Fix lemezegység partíciójának kijelölése.
FIND	Megadott karaktersorozat keresése szöveges file-ban.
FORMAT	Lemezek formázása, inicializálása, a DOS rezidens file-jainak elhelyezése.
GRAFTABL	A grafikus jelkészlet betöltése a memóriába.
GRAPHICS	A grafikus nyomtatás lehetőségének beállítása (színes-grafikus adapter szükséges).
JOIN	Két fizikailag különböző lemezegység összekapcsolása egy logikai egységgé.

Parancsok	Funkció
KEYB	A DOS 3.30-as verziójának billentyűzet-kezelő programja.
KEYBxx	Az adott ország nemzeti billentyűzetét kezelő rutin betöltése.
LABEL	Lemez kötetazonosítójának (címkéjének) megadása.
MKDIR	A fastruktúrájú tartalomjegyzékben altartalomjegyzék létrehozása. *Rövidített formája*: MD.
MODE	A képernyő, nyomtató üzemmódjának beállítása, és az adatátviteli vonalak inicializálása.
MORE	File-ok képernyő méretű oldalakra tagolása.
NLSFUNC	Kiterjesztett ország-függő információk és bizonyos kód-lap kezelő (pl. CHCP) eljárások engedélyezése.
PATH	A bejárási utak kijelölése a fastruktúrájú tartalomjegyzékben. (Csak .EXE, .COM és .BAT kiterjesztés jó!)Környezet paraméter is (SET parancs).
PRINT	A nyomtatás várakozási sorának vezérlése (háttér nyomtatás).
PROMPT	A PROMPT karaktersor beállítása.
RECOVER	File-ok ismételt előállítása. A sérült lemezterületen kívül, vagy a tartalomjegyzék meghibásodása esetén új bejegyzések elkészítése a fileleíró táblázat (FAT) alapján.
RENAME	File-ok átnevezése. *Rövidített formája*: REN.
REPLACE	A cél tartalomjegyzék file-jainak helyettesítése a forrás file-okkal. (Új file nem keletkezik a cél lemezen).
RESTORE	A fix lemez állományainak visszatöltése a tartalék (backup) lemezekről.
RMDIR	Üres altartalomjegyzék törlése. *Rövidített formája*: RD.
SELECT	DOS installálása egy új (hajlékony) lemezen, adott billentyűzet rutinnal és dátum formával.
SET	A program-környezetben paraméterek elhelyezése, illetve a környezet-paraméterek lekérdezése.
SHARE	File megosztási környezet installálása.
SORT	Szöveges file-ok rendezése (növekvő vagy csökkenő sorrendben a rekordok megadott pozíciójától).
SUBST	Adott útra vagy lemezegységre másik lemezegységként hivatkozhatunk.
SYS	A DOS rezidens file-jainak elhelyezése egy lemezen.
TIME	A rendszer-idő beállítása, lekérdezése.
TREE	Tartalomjegyzék fastruktúrájának kilistázása.
TYPE	File-ok tartalmának kiíratása (szöveges/text típus esetén).
VER	A DOS verziójának lekérdezése.
VERIFY	A "lemezre írás-ellenőrzés" kapcsoló beállítása, állapotának lekérdezése.
VOL	Lemez kötetazonosítójának lekérdezése.
XCOPY	File-ok vagy teljes altartalomjegyzék másolása egy másik lemezre vagy altartalomjegyzékbe.

Batch alparancsok	Funkció
CALL	Parancs-file-ból másik parancs-file meghívása.
ECHO	A parancsok kijelzésének be-, illetve kikapcsolása.
FOR	DOS parancsok többszöri végrehajtása.
GOTO	A feldolgozás a címkével megjelölt sortól folytatódik.
IF	Feltételes parancs-végrehajtás.
PAUSE	A parancsok feldolgozásának felfüggesztése.
REM	Megjegyzések (REMarks) elhelyezése a parancs-kötegben.
SHIFT	A parancs-paraméterek léptetése (több, mint tíz paraméter esetén).

Konfigurációs parancsok	Funkciók
BREAK	A BREAK billentyű figyelésének beállítása.
BUFFERS	Az I/O műveletekhez használt lemez pufferek száma.
COUNTRY	A használt dátum és idő kijelzési formátum megadása.
DEVICE	A DOS által használandó perifériakezelők megadása.
FCBS	Az egyszerre megnyitható FCB-k (file controll blokk) száma.
FILES	Az egyidőben megnyitott file-ok maximális száma.
LASTDRIVE	A maximálisan használható lemezegység betűjele.
SHELL	A felhasználói parancs-processzor azonosítója.
STACKS	A hardware megszakítások kezeléséhez szükséges veremméret megadása.

C. Függelék

Parancsok, alparancsok formája

Parancs	Típus	Forma
APPEND	K	**APPEND** *d:út*[;[*d:*]*út*...] **APPEND** [/X][/E] **APPEND** [;]
ASSIGN	K	**ASSIGN** [*x*=*y*[...]]
ATTRIB	K	**ATTRIB** [+R][-R][+A][-A][*d:*][*út*]*file-név*[*.kit*][/S]
BACKUP	K	**BACKUP** [*d:*][*út*][*file-név*[*.kit*]][*d:*][/S][/M][/A][D:*hh-nn-éé*]
BACKUP (MS DOS)	K	**BACKUP** [*d:*][*út*] [*d:*][/S][/M][/A][/P] [/D:*dátum*][/T:*idő*][/L:*file-név*[*.kit*]]
BACKUP (3.30)	K	**BACKUP** [*d:*][*út*][*file-név*[*.kit*]][*d:*][/S][/M][/A] [/D:*dátum*][/T:*idő*][/F][/L[:[*d:*][*út*]*file-név*[*.kit*]]]
BREAK	B	**BREAK** [ON|OFF]
CHCP	K	**CHCP** [*kl*]
CHDIR	B	**CHDIR** [*d:*][*út*] **CD** [*d:*][*út*]
CHKDSK	K	**CHKDSK** [*d:*][*út*][*file-név*[*.kit*]][/V][/F]
CLS	B	**CLS**
COMMAND	K	**COMMAND** [*d:*][*út*][/P][/C *karaktersorozat*][/E:*xxxx*]
COMP	K	**COMP** [*d:*][*út*]*file-név*[*.kit*][*d:*][*út*][*file-név*[*.kit*]]
COPY	B	**COPY** [/A][/B][*d:*][*út*]*filenév*[*.kit*][/A][/B] [*d:*][*út*][*file-név*[*.kit*]][/A][/B][/V] **COPY** [/A][/B][*d:*][*út*]*filenév*[*.kit*][/A][/B] [+[*d:*][*út*]*file-név*[*.kit*][/A][/B]...] [[*d:*][*út*]*file-név*[*.kit*][/A][/B][/V]
CTTY	B	**CTTY** *egységnév*
DATE	B	**DATE** [*hh-nn-éé*] | [*éé-hh-nn*] | [*nn-hh-éé*]
DEL	B	**DEL** [*d:*][*út*]*file-név*[*.kit*]
DIR	B	**DIR** [*d:*][*út*][*filenév*[*.kit*]] [/P][/W]
DISKCOMP	K	**DISKCOMP** *d: d:*[/1][/8]
DISKCOPY	K	**DISKCOPY** *d: d:*[/1]
ERASE	B	**ERASE** [*d:*][*út*]*file-név*[*.kit*]
EXE2BIN	K	**EXE2BIN** [*d:*][*út*]*file-név*[*.kit*][*d:*][*út*][*file-név*[*.kit*]]
EXIT	B	**EXIT**
FASTOPEN	K	**FASTOPEN** *d:*[*nnn*]...
FDISK	K	**FDISK**
FIND	K	**FIND** [/V][/C][/N] *karaktersorozat*[[*d:*][*út*]*file-név*[*.kit*]...]

FORMAT	K	**FORMAT** [*d:*][/S][/1][/8][/N][/B][/V][/N:*xx*][/T:*yy*]
GRAFTABL	K	**GRAFTABL** [437|860|863|865|/STATUS]
GRAPHICS	K	**GRAPHICS** [*nyomtató-típus*][/R][/B][/LCD]
JOIN	K	**JOIN** [*d1: d2:*[*út*]]|[*d:*/D]
KEYB	K	**KEYB** [*xx*[,[*yyy*],[[*d:*][*út*]*file-név*[.*kit*]]]]]
KEYBxx	K	**KEYBxx**
LABEL	K	**LABEL** [*d:*][*címke*]
MKDIR	B	**MKDIR** [*d:*]*út*
		MD [*d:*]*út*
MODE	K	**MODE LPT#:**[*n*][,[*m*][,P]]
		MODE *n*
		MODE [*n*],*m*[,T]
		MODE COM*n:baud*[,*paritás*[,*adatbitek*[,*stopbitek*[,P]]]]
		MODE *egység* **CODEPAGE PREPARE =** ((*kl*)[*d:*][*út*]*file-név*[.*kit*])
		MODE *egység* **CODEPAGE PREPARE =** ((*kl-lista*)[*d:*][*út*]*file-név*[.*kit*])
		MODE *egység* **CODEPAGE SELECT =** *kl*
		MODE *egység* **CODEPAGE** [/STATUS]
		MODE *egység* **CODEPAGE REFRESH**
MORE	K	**MORE <** [*d:*][*út*]*file-név*[.*kit*]
NLSFUNC	K	**NLSFUNC** [[*d:*][*út*]*file-név*[.*kit*]
PATH	B	**PATH** [*d:*]*út*[;[*d:*]*út*...]
		PATH [;]
PRINT	K	**PRINT** [/D: *egységnév*][/B: *puffernméret*][/U: *foglalt*][/M: *nyomtat*] [/S:*várakozik*][/Q: *sorméret*][/C][/T][/P][*d:*][*út*][*file-név*.[*kit*]...]
PROMPT	K	**PROMPT** *prompt jelek*
		$ - $ karakter
		t - rendszer idő
		d - rendszer dátum
		p - aktuális tartalomjegyzék
		v - a DOS verziója
		n - aktuális meghajtó
		g - > karakter
		l - < karakter
		b - ' karakter
		q - = karakter
		h - kurzor visszaléptetése, megelőző karakter törlése
		e - ESCape karakter
		_ - CR/LF a képernyő következő sorának elejére kerül a kurzor
RECOVER	K	**RECOVER** [*d:*][*út*]*file-név*[.*kit*]
		RECOVER [*d:*]

RENAME	B	**RENAME** [*d:*][*út*]*file-név*[.*kit*]*file-név*[.*kit*]*file-név*[.*kit*] **REN** [*d:*][*út*]*file-név*[.*kit*]
REPLACE	K	**REPLACE** [*d:*][*út*]*filenév*[.*kit*][*d:*][*út*][/A][/P][/R][/S][/W]
RESTORE	K	**RESTORE** *d:* [*d:*][*út*]*filenév*[.*kit*][/S][/P]
RESTORE (3.30)	K	**RESTORE** *d:* [*d:*][*út*]*file-név*[.*kit*][/S][/P] [/B:*dátum*][/A:*dátum*][/M][/N][/L:*idő*][/E:*idő*]
RMDIR	B	**RMDIR** [*d:*]*út* **RD** [*d:*]*út*
SELECT	K	**SELECT** [[A: | B:] *d:*[*út*]] *xxx yy*
SET	B	**SET** [*név* = [*paraméter*]]
SHARE	K	**SHARE** [/F:*fileméret*][/L:*zárások*]
SORT	K	**SORT** [/R][/ + *n*]
SUBST	K	**SUBST** [*d: d1:út*] | [*d:*/D]
SYS	K	**SYS** *d:*
TIME	B	**TIME** [*óó:pp:mm.xx*]
TREE	K	**TREE** [*d:*][/F]
TYPE	B	**TYPE** [*d:*][*út*]*filenév*[.*kit*]
VER	B	**VER**
VERIFY	B	**VERIFY** [ON | OFF]
VOL	B	**VOL** [*d:*]
XCOPY	K	**XCOPY** [*d:*][*út*][*filenév*[.*kit*]][*d:*][*út*][*filenév*[.*kit*]] [/A][/D][/E][/M][/P][/S][/V][/W]

Batch alparancs	**Típus**	**Forma**
CALL	B	**CALL** [*d:*][*út*]*file-név*
ECHO	B	**ECHO** [ON | OFF | *üzenet*]
FOR	B	**FOR** %%*változó* IN(*értékek halmaza*) DO *parancs*
GOTO	B	**GOTO**[:]*címke*
IF	B	**IF** [NOT] *feltétel parancs* A feltétel alakja: **ERRORLEVEL** *szám* *sztring 1* = = *sztring 2* **EXIST** [*d:*][*út*]*file-név*[.*kit*]
PAUSE	B	**PAUSE** [*üzenet*]
REM	B	**REM** [*megjegyzés*]
SHIFT	B	**SHIFT**

Konfigurációs parancsok	**Forma**
BREAK	**BREAK = ON|OFF**
BUFFERS	**BUFFERS** = *xx* (1< = xx< = 99)
COUNTRY	**COUNTRY** = *xxx*
COUNTRY	**COUNTRY** = *xxx*,[*yyy*]
(3.30)	**COUNTRY** = *xxx*,[*yyy*],[*d:*][*út*]*file-név*[.*kit*]
DEVICE	**DEVICE** = [*d:*][*út*]*filenév*[.*kit*][*paraméterek*]
	DEVICE = DRIVER.SYS /D:*ddd*[**/T:***tt*][**/S:***sss*][**/H:***hh*][**/C**][**/W**][**/F:***f*]
	DEVICE = VDISK.SYS [*bbb*][*sss*][*ddd*][**/E**[:*m*]]
	DEVICE = ANSI.SYS
	DEVICE = DISPLAY.SYS CON[:] = (*típus* [,[*hkl*][,*n*]])
	DEVICE = DISPLAY.SYS CON[:] = (*típus* [,[*hkl*][,(*n*,*m*)]])
	DEVICE = PRINTER.SYS LPT#[:] = (*típus* [,[*hkl*][,*n*]])
	DEVICE = PRINTER.SYS LPT#[:] = (*típus* [,[(*hkl1*,*hkl2*,...)][,*n*]])
FCBS	**FCBS** = *m*,*n*
FILES	**FILES** = *xx* (1< = xx< = 99)
LASTDRIVE	**LASTDRIVE** = *x*
SHELL	**SHELL** = [*d:*][*út*]*filenév*[.*kit*][*paraméter1*][*paraméter2*]
STACKS	**STACKS** = *n*,*s*

D. Függelék

File-név kiterjesztések

A PC DOS által használt file-név kiterjesztések és rövid magyarázatuk:

Kiterjesztés	**Magyarázat**
.BAK	a file korábbi változata
.BAS	BASIC forrásprogram
.BAT	parancs-file (batch file)
.COM	végrehajtható (gépi kódú) program közvetlenül betölthető formában
.CPI	kód-lapokra vonatkozó információt tartalmazó file
.EXE	végrehajtható (gépi kódú) program áthelyezhető formában
.HEX	hexadecimális karaktereket tartalmazó file
.LIB	programkönyvtár (library)
.MAP	memóriatérkép a globális szimbólumokról
.OBJ	tárgy (objekt) program
.SYS	periféria-kezelő vagy annak tartozéka
.TMP	ideiglenes munka file
.$$$	ideiglenes munka file

E. függelék

Eszköznevek listája

A DOS parancsokban, ahol a file-specifikációt kell írnunk, az alábbi eszköznevek használata megengedett (Az eszköznevek utáni kettőspont használata nem kötelező.):

Eszköznév	**Input vagy output**	**Magyarázat**
AUX:	input és output	az első aszinkron adapter
COM1:	input és output	az első aszinkron adapter
COM2:	input és output	a második aszinkron adapter
COM3:	input és output	a harmadik aszinkron adapter (DOS 3.30-tól)
COM4:	input és output	a negyedik aszinkron adapter (DOS 3.30-tól)
CON:	input és output	input esetén billentyűzet, output esetén a képernyő
LPT1:	output	az első párhuzamos nyomtató
LPT2:	output	a második párhuzamos nyomtató
LPT3:	output	a harmadik párhuzamos nyomtató
NUL:	input és output	nemlétező (dummy) eszköz. Olvasásakor file vége jelet kapunk, írásakor tényleges átvitel nem történik
PRN:	output	az első párhuzamos nyomtató

%%, 113
%, 113
.BAT, 105
.COM, 18
.EXE, 18
< Ctrl-PtrSc >, 32
< F1 >, < F2 >...< F10 >, 29
< F6 >, 25
aktuális blokk, 245
aktuális lemezegység, 38
aktuális meghajtó, 34
aktuális prompt, 202
aktuális rekord, 244
aktuális tartalomjegyzék, 34, 61
alparancs, 105
altartalomjegyzék, 60, 220
ANSI.SYS, 173, 174
append, 139
APPEND, 79
ASCII rendezési súly, 188
ASCII érték, 188
ASCII-kód, 41
assemble, 151
ASSIGN, 94
ATTRIB, 54
attributum byte, 246
attributum, 46, 55
AUTOEXEC.BAT, 23, 107
auxiliary carry, 146
AX, 143, 146
BACKUP, 85, 111
Basic I/O System, 236
batch üzemmód, 12
batch-file, 105
befejezési kód, 110
BIOS paraméter blokk, 215, 228
BIOS, 14, 17, 236
blokkos berendezések, 212
boot record, 17
boot szektor, 214
boot, 204, 205, 214
BP, 143, 146, 205
BP:SI, 256
Break, 171
BREAK, 95
breakpoint, 154
BUFFERS, 172
BX, 143, 146
CALL, 114
carry, 146
CHCP (kódlap váltás), 193
CHDIR, 34
CHDIR, 61
CHKDSK, 75
cilinder, 82
CLS, 92
command processor, 65
COMMAND, 90
COMMAND.COM, 18, 202
COMP, 48
COMPARE, 48, 149
CONFIG.SYS, 171
COPY, 41, 131
COUNTRY, 172
CS, 143, 146, 205
Ctrl-Break, 252
CTTY, 190
CX, 143, 146
DATE, 92
DEBUG, 27, 141
delete, 125
Device Header Control Blokk, 254
device header, 223
DEVICE, 173
DF, 143
DI, 146, 205
DIR, 70
direction, 146
directory, 11, 12, 220
disassemblálás, 141, 153
Disk Transfer Area, 143
DISKCOMP, 74
DISKCOPY, 24, 72
DISPLAY.SYS, 143
DISPLAY.SYS-kódlap, 177
DOS, 11
driver, 223
DRIVER.SYS, 173, 178
DS, 143, 146, 205
DSALLOCATION, 169
DTA, 279, 287
dump, 145

duplaszó, 245
DX, 143, 146
ECHO, 108
EDLIN, 27, 115
egy pont (.), 71
egység leíró tábla, 228
elérési út, 35
elsődleges DOS partíció, 81
elsődleges és másodlagos file-azonosító, 49
end, 128
enter, 148
environment, 236
ERASE, 51
ERRORLEVEL, 110
ES, 143, 146, 205
EXE2BIN, 191
EXIST, 110
EXIT, 91
FASTOPEN, 59
FAT (File Allocation Table), 21
FCB, 273
FCB-orientált eljárás, 243
FCBS, 181
FDISK, 25, 81
file, 11
file allocation table (FAT), 75
file attributum, 245
file átnevezés, 52
file azonosító, 35, 36
File Control Block, 156
file kezelés, 243, 245
file kiterjesztés, 222
file kontroll blokk (FCB), 181
file leíró blokk, 243
file leíró tábla, 214, 217
file megosztási környezet, 55
file méret, 248, 250
file-név kiterjesztés, 71
file-név, 17, 35, 248
file-pointer, 251
file-szám, 243, 251
file-vezérlő blokk, 156
file-vég jel, 49
FILES, 182
fill, 147
FIND, 186
fix lemez, 85
flag, 143
FOR, 112
FORMAT, 26, 65
formattál, 65
főtartalomjegyzék, 33, 39
funkció, 251, 260
funkcióhívás, 211, 243
futtatható file, 161
függvényhívás, 205
globális file, 11
globális file-név, 36, 42
go, 154
GOTO, 112
GRAFTABL, 102
GRAPHICS, 103
gyökér tartalomjegyzék, 33
handler, 228
hexadecimális, 160
hibakeresés, 11
HIGH, 167, 169
IBMBIO.COM, 17, 52, 65
IBMBIO.SYS, 202
IBMDOS.COM, 17, 52, 65
IBMDOS.SYS, 202
IF, 110
inicializálás, 228
input, 160
insert, 119
INT, 209
interaktív üzemmód, 162
interrupt vektor, 102
interrupt vektortábla, 210, 281
interrupt, 146, 252
IP, 143, 146, 205
JOIN, 78
karakteres berendezés, 212, 260
karakteres periféria, 231
karaktersorozat, 132
KEYBGR, 25
KEYBxx, 101
két pont (..), 71
kiegészítő DOS partíció, 81
kilépési cím, 252
kilépési rutin, 253
kiterjesztés, 35, 244

konfigurációs file, 171
konfigurációs parancsok, 171
könyvtár-file, 162
kötegelt feldolgozás, 11
kötetazonosító, 65
LABEL, 72
LASTDRIVE, 183
lemez-file, 167
lemezegység, 86
lemezkezelés, 213
lemezátviteli terület, 246
LINE, 170
LINK, 15, 161
list, 120
load, 157
logikai egység, 82
logikai rekordméret, 245
logikai sorszám, 213
MAP, 167
map-file, 163, 166
megszakítás, 207, 252, 281
megszakítási pont, 154, 207
megszakítási rendszer, 205
meleg újraindítás, 203
memóriablokk, 241
memória leíró blokk (MCB), 241
memórialemez, 183
MKDIR, 60
MODE, 96
MORE, 186
move, 129, 150
name, 156
NLSFUNC, 193
old, 170
opció, 161
ország-függő információk, 193
overflow, 146
page, 122
paragrafus, 241
parancs-file, 12, 105
parancs-processzor, 18, 183
parancsblokk, 225
parity, 146
partíció, 82, 214
partíciós tábla, 214
PATH, 63
PAUSE, 109
PC, 143
periféria, 260
piping, 12
PRINT, 57
PRINTER.SYS, 173, 179
proceed, 155
programkörnyezet, 193, 236
programszegmens prefix (PSP), 143, 237
prompt, 143
PROMPT, 189
PSP (Program Segmens Prefix), 193, 236
quit, 128, 151
random, 248, 251, 279
random írás, 247, 279
random olvasás, 247, 279
RECOVER, 53
register, 146, 205
rejtett (hidden) file, 52, 246
rekord abszolút sorszáma, 247
relatív cím, 144
REM, 108
RENAME, 52
replace, 135
REPLACE, 47
RESTORE, 88
rezidens program, 241, 258
RMDIR, 62
sáv (track), 18, 213
search, 132, 150
SELECT, 68
SET, 193
SHARE, 55
SHELL, 183
SHIFT, 113
SI, 143, 146, 205
sorszám, 120
SORT, 187
SP, 143, 146
spooling, 57
SS, 143, 146, 207
STACK, 168, 183
standard I/O átirányítása, 184
SUBST, 77
SYS, 69
százalék, 106

szegmenscím, 153, 241
szegmensregiszter, 143, 206
szegmesszám, 170
szektor, 213
szekvenciális, 246, 251, 276
szó, 244
szöveg-file, 49
tartalomjegyzék, 11, 12, 35, 220
tárgy-file, 161
TIME, 93
trace, 155
track, 18
transfer, 137
TREE, 63
TYPE, 56
unassemble, 153
válasz-file, 164
VDISK.SYS, 173, 180
VER, 94
veremhasználat, 183
veremmutató, 143
VERIFY, 77
VOL, 71
volume label, 26
warm reset, 205
winchester, 213
write, 138, 158
XCOPY, 45

IRODALOMJEGYZÉK

Ashley, R., Fernandez, J. N.:
PC DOS: Using the IBM PC Operating System
J. Wiley & Sons, Inc., 1983.

Bradley, D. J.:
Assemblly Language Programming for the IBM Personal Computer
Prentice-Hall, 1984.

Disk Operating System
(Version 2.00, 3.00, 3.10, 3.20, 3.30)
(IBM PC Computer Language Series)
IBM Corp., 1984-1987.

Graham, L. J., Field, T.:
Your IBM PC
A Guide to the IBM PC (DOS 2.00) and XT
ed.: Osborne/McGraw-Hill, Berkeley, California

IBM Personal Computer XT
Technical reference
IBM Corp., 1983.

IBM Personal Computer AT
Technical reference
IBM Corp., 1984.

IBM DOS Technical Reference
(Version 3.00, 3.10, 3.20, 3.30)
IBM Corp., 1984-1987.

King, R. A.:
The IBM PC DOS Handbook
Berkeley, California SYBEX, Inc., 1983.

Microsoft Macro Assembler for the MS-DOS Operating System
Reference Manual
Microsoft Corporation 1984, 1985.

Microsoft
MS-DOS Operating System
Programmer's reference Manual
Microsoft Corp., 1984-1987.

Morse, S. P.:
The 8086/8088 Primer
Hayden Book Co., 1982.

Norton, P.:
Inside the IBM PC
Robert J. Brady Co., 1983.

Norton, P.:
MS-DOS and PC-DOS User's Guide
Robert J. Brady Co., 1984.

Reector, R. - Alexy, G.:
The 8086 Book
Osborne/McGraw-Hill, 1980.

Sargent, M. III - Shoemaker, R. L.:
The IBM Personal Computer From the Inside Out
Addison-Wesley Publishing Co., 1984.

Szilassy Bertalan:
Az i8086 mikroprocesszor 1. kötet Utasításkészlet
LSI ATSz, Bp., 1983.

SzKI SCI-L Proper 16/A professzionális személyi számítógép
Operációs rendszer kézikönyv
SzKI SCI-L, Bp., 1983.

SzKI SCI-L PROPOS-16
Operációs rendszer és a WINCHESTER DISK használata
Felhasználói leírás
SzKI SCI-L, Bp., 1984.

Top Wiew
Programmer's Toolkit
IBM Corp., 1984.

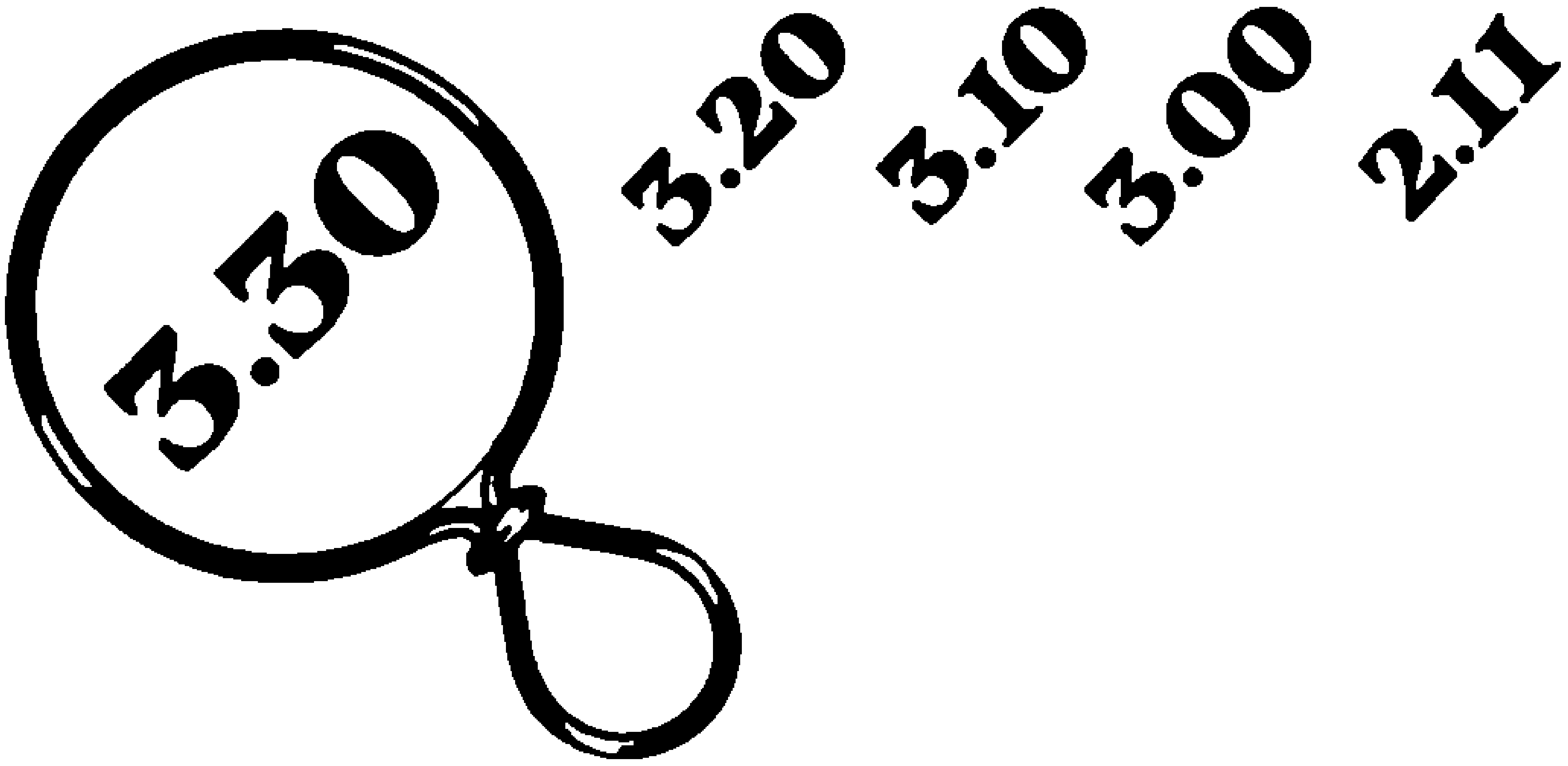
IBM PC DOS

III. PC DOS
PROGRAMOZÓI SEGÉDLET

3.30
3.20
3.10
3.00
2.11

IBM PC DOS

III. PC DOS PROGRAMOZÓI SEGÉDLET


3.30
3.20
3.10
3.00
2.11

ALKALMAZÁSTECHNIKAI TANÁCSADÓ SZOLGÁLAT
BUDAPEST, 1990

Kósza Zsolt

Szerkesztette: Dr. Úry László III.

Készítette: Orosz Judit
Baki Zoltán
Pajor Gábor

Lektorálta: Éltető László

Kiadó: LSI Alkalmazástechnikai Tanácsadó Szolgálat
Felelős kiadó: dr. Kovács Magda
Témafelelős: Székely László
Szedés: LSI ATSz Info Osztály
Technikai szerkesztő: Nagy Olivér
Engedélyszám: 57152
ISBN összkötet: 963 576 004/3
ISBN I. kötet: 963 576 0005/1
ISBN II. kötet: 963 576 006/x
ISBN III. kötet: 963 576 007/8
AGRO – PRINT Kft, Gyál 90-67/3
Felelős vezető: Tóth Antal

Előszó

Ez a kézikönyv az IBM PC XT/AT-kompatibilis mikroszámítógépek egyik leghatékonyabb programfejlesztő eszközének, a macro assembler-nek használatához szükséges ismereteket foglalja össze.

A DOS-kézikönyv csak a megszakítások és funkcióleírások megértéséhez szükséges legfontosabb információkat foglalta össze a 8086 mikroprocesszor-családról, jelen könyv 2. fejezete részletesen ismerteti a processzor felépítését és az utasítások struktúráját.

A 3.-5. fejezetek a forrásnyelvű assembly programok felépítésével foglalkoznak, részletesen tárgyalják az assembly utasításokat és használható direktívákat.

A további fejezetek magáról a MASM macro assembler fordítóprogram, illetve a kapcsolódó segédprogramok használatát mutatják be. Ezek közé tartozik a LIB programkönyvtár-kezelő program, a LINK szerkesztő, a CREF keresztհivatkozás-listát készítő program, valamint a MAKE és a programok nyomkövetését biztosító szimbólum hibakereső program, a SYMDEB.

A függelék az assembler és a szerkesztő hibaüzeneteit tartalmazza.

A Kiadó

Tartalomjegyzék

1. BEVEZETÉS 10

2. A MIKROPROCESSZOR-CSALÁD ARCHITEKTÚRÁJA 11
2.1 A 8088 CPU blokkdiagramja 11
2.2 Regiszterek 12
2.3 Tárcímzés szegmentálással 12
2.4 Adatcímzési módok 13
2.5 Az utasításkészlet fő csoportjai 15
2.5.1 Adatátviteli utasítások 15
2.5.2 Aritmetikai utasítások 16
2.5.3 Logikai utasítások 16
2.5.4 Karaktersorozat-kezelő utasítások 18
2.5.5 Vezérlésátadó utasítások 19
2.5.6 Processzorvezérlő utasítások 22

3. ADATTÍPUSOK 23
3.1 Karakterkészlet 23
3.2 Numerikus konstansok 23
3.2.1 Egész számok 23
3.2.2 Valós számok 24
3.2.3 Kódolt valós számok 24
3.2.4 Tömörített decimális számok 24
3.3 Karakter- és karaktersorozat-konstansok 25
3.4 Azonosítók 25
3.5 Operandusok 26
3.5.1 Konstans operandusok 26
3.5.2 Direkt címmel elérhető operandusok 26
3.5.3 Relatív címmel elérhető operandusok 27
3.5.4 Számláló operandus 27
3.5.5 Regiszter-operandusok 27
3.5.6 Bázis-relatív operandusok 28
3.5.7 Indexelt operandusok 29
3.5.8 Bázis-relatív indexelt operandusok 29
3.5.9 Struktúra-operandusok 30
3.5.10 Rekord-operandusok 30
3.5.11 Rekordmező-operandusok 30
3.6 Az operandusokból előállítható kifejezések 31
3.6.1 Aritmetikai műveletek 31
3.6.2 Léptető műveletek 32
3.6.3 Logikai műveletek 32
3.6.4 Indexelés 32
3.6.5 A PTR művelet 34
3.6.6 A címkiszámítás viszonyítási alapjának megváltoztatása 34

3.6.7 A SHORT művelet ... 35
3.6.8 A THIS művelet ... 35
3.6.9 A HIGH és a LOW művelet ... 36
3.6.10 A SEG művelet ... 36
3.6.11 Az OFFSET művelet ... 36
3.6.12 A TYPE művelet ... 37
3.6.13 A .TYPE művelet ... 37
3.6.14 A LENGTH művelet ... 38
3.6.15 A SIZE művelet ... 39
3.6.16 A WIDTH művelet ... 39
3.6.17 A MASK művelet ... 39
3.6.18 A kifejezés értékének kiszámítása és a precedencia ... 40
3.6.19 A memória-operandusok típusának ellenőrzése ... 40

4. A MIKROPROCESSZOR-CSALÁD ASSEMBLY UTASÍTÁSAI ... 41
4.1 Az utasítások leírásában használt rövidítések ... 41
4.2 A 8086 és a 8088 utasításkészlete ... 42
4.3 A 8087 utasításkészlete ... 87
4.3.1 A 8087-es numerikus adatprocesszor felépítése ... 87
4.4 A 80186 és a 80286 utasításkészlete ... 93
4.5 A 80287 utasításkészlete ... 95

5. PROGRAMOZÁS MAKRO-ASSEMBLY NYELVEN ... 96
5.1 A makró-assembly programok felépítése ... 96
5.1.1 A makró-assembly utasítások általános alakja ... 96
5.1.2 Az utasításkészletet beállító direktívák ... 96
5.1.3 A forrás-modulok és az END direktíva ... 97
5.1.4 A SEGMENT és az ENDS direktíva ... 98
5.1.5 Az ORG direktíva ... 99
5.1.6 Az EVEN direktíva ... 99
5.1.7 A PROC és az ENDP direktíva ... 100
5.1.8 A GROUP direktíva ... 100
5.1.9 Az ASSUME direktíva ... 101
5.1.10 A megjegyzések és a COMMENT direktíva ... 101
5.1.11 Példák a makró-assembly programok felépítését meghatározó direktívák használatára ... 102
5.2 Deklarációk ... 103
5.2.1 Adatdeklarációk ... 103
5.2.2 Címkedeklarációk ... 108
5.2.3 Szimbólumdeklarációk ... 108
5.2.4 Struktúrák deklarálása ... 110
5.2.5 Rekordok deklarálása ... 113
5.2.6 Előre-hivatkozások ... 115
5.2.7 Globális deklarációk ... 116
5.3 Makrók definiálása és hívása ... 119
5.3.1 A MACRO és az ENDM direktíva ... 119

5.3.2 Makróhívások 121

5.3.3 Részkaraktersorozat megadása makró-paraméterrel 121

5.3.4 A szöveg-konstans jele 123

5.3.5 A karakter-konstans jele 123

5.3.6 A kifejezés jele 123

5.3.7 A PURGE direktíva - makródefiníciók törlése 124

5.3.8 A LOCAL direktíva - egyedi azonosítók létrehozása makrókban 124

5.3.9 A REPT és az ENDM direktíva - utasításcsoport ismétlése 125

5.3.10 Az IRP és az ENDM direktíva - utasításcsoport ismétlése paraméterhelyettesítéssel 126

5.3.11 Az IRPC és az ENDM direktíva 127

5.3.12 Az EXITM direktíva 128

5.3.13 Makró-megjegyzések 128

5.3.14 Feltételes fordítás 129

3.14.1 Az IF és az IFE direktíva 130

3.14.2 Az IF1 és az IF2 direktíva 130

3.14.3 Az IFDEF és az IFNDEF direktíva 130

3.14.4 Az IFB és az IFNB direktíva 131

3.14.5 Az IFIDN és az IFDIF direktíva 131

5.4 A makró-assembler file-kezelésének vezérlése 132

5.4.1 Az INCLUDE direktíva 132

5.4.2 A TITLE direktíva 133

5.4.3 A SUBTITLE direktíva 133

5.4.4 A PAGE direktíva 133

5.4.5 A .LIST és a .XLIST direktíva 134

5.4.6 A .LALL, a .XALL és a .SALL direktíva 134

5.4.7 A .SFCOND, a .LFCOND és a .TFCOND direktíva 135

5.4.8 A .RADIX direktíva 135

5.4.9 A .CREF és a .XCREF direktíva 136

5.4.10 A .NAME direktíva 136

5.4.11 A %OUT direktíva 137

5.5 Magasszintű nyelvből hívott Assembly rutinok 137

6. A MASM FORDÍTÓ HASZNÁLATA 138

6.1 Bevezetés 138

6.2 Forrás file fordítása 138

6.3 A MASM kapcsolói 139

6.3.1 /D, az első menet listája 139

6.3.2 /ML, kis-nagybetűk megkülönböztetése a lokális szimbólumokban 139

6.3.3 /MX, kis-nagybetűk megkülönböztetése a globális szimbólumokban 139

6.3.4 /X, a feltételes fordítások hamis ágának listázása 140

6.3.5 /R, kódképzés az aritmetikai processzor számára 140

6.3.6 /E, kódképzés a lebegőpontos emulátor számára 140

6.3.7 /A, szegmensek abc sorrendben 140

6.4 A lista-file 140

6.4.1 A forrássorok 141

6.4.2 A makrótáblázat 141
6.4.3 A struktúra-, rekordtáblázat 141
6.4.4 A szegmens-, csoporttáblázat 142
6.4.5 A szimbólumtáblázat 142
6.4.6 Az első menet listája 143

7. PROGRAMKÖNYVTÁR-KEZELÉS A LIB SEGÉDPROGRAMMAL 144
7.1 A programkönyvtár és kezelése 144
7.2 A LIB funkciói 144
7.2.1 A LIB hívása parancs-sorral 146
7.2.2 A LIB használata párbeszédes módban 146
7.2.3 LIB parancs-sorozat használata file-ból (válasz-file) 148
7.3 Új könyvtár létrehozása 149
7.4 A lapméret átállítása (PAGESIZE) 149
7.5 A könyvtár konzisztenciájának ellenőrzése 150
7.6 A keresztreferencia-lista és készítése 150

8. FUTTATHATÓ PROGRAM ELŐÁLLÍTÁSA - Programszerkesztés (LINK) 151
8.1 A LINK indítása 151
8.1.1 Programszerkesztés PC DOS parancs-sorral 152
8.1.2 Programszerkesztés interaktív üzemmódban 154
8.1.3 Programszerkesztés előre összeállított válasz-file-val 154
8.1.4 Könyvtárak elérési útjainak megadása 155
8.1.5 A map file szerkezete 155
8.1.6 Ideiglenes lemez-file - VM.TMP 156
8.2 A LINK opciói 157
8.2.1 Megszakítás a szerkesztés alatt - PAUSE 157
8.2.2 Public szimbólumok táblájának előállítása - MAP 158
8.2.3 A verem-méret meghatározása - STACK 158
8.2.4 A program helyigényének maximuma - CPARMAXALLOC 186
8.2.5 A kezdőcím maximumra állítása - HIGH 160
8.2.6 Helyfoglalás az adatcsoportnak - DSALLOCATE 160
8.2.7 Forrás-sorok kezdőcímének kiírása - LINENUMBERS 160
8.2.8 Kis- és nagybetűk megkülönböztetése - NOIGNORECASE 160
8.2.9 A könyvtár-alapfeltételezés törlése - NODEFAULTLIBRARYSEARCH 161
8.2.10 Csoportok eltávolítása a programból - NOGROUPASSOCIATION 116
8.2.11 Megszakítás overlay esetén - OVERLAYINTERRUPT 161
8.2.12 A szegmensek maximális számának meghatározása - SEGMENTS 162
8.2.13 Az MS-DOS szegmens-elrendezés alkalmazása - DOSSEG 162
8.3 A LINK működése 162
8.3.1 A szegmensek címhatárra való igazítása 163
8.3.2 A szegmensek kezdőcímének meghatározása 163
8.3.3 A szegmensek átrendezése 163
8.3.4 Kombinált szegmensek 164
8.3.5 Szegmens-csoportok 164
8.3.6 Címfeloldás 164

8.3.7 A betöltés sorrendjének vezérlése 165

9. PROGRAMOK KARBANTARTÁSA A MAKE PROGRAM SEGÍTSÉGÉVEL 167
9.1 Bevezetés 167
9.2 A MAKE használata 167
9.3 MAKE leíró file-ok létrehozása 167
9.4 A MAKE futtatása 168
9.5 MAKE opciók 168
9.6 Makró definíciók 169
9.7 Példa a MAKE használatára 169

10. SZIMBÓLIKUS NYOMKERESŐ ÉS HIBAKÖVETŐ PROGRAM (SYMDEB) 170
10.1 Bevezetés 170
10.2 A SYMDEB indítása 170
10.2.1 Indítás program file-lal 171
10.2.2 Indítás szimbólumokkal 171
10.2.3 Paraméterátadás a betöltendő programnak 171
10.2.4 Indítás file nélkül 171
10.2.5 Szimbólum file készítése (MAPSYM) 171
10.3 A kontroll karakter funkciója 172
10.3.1 Parancs megszakítása 172
10.3.2 Parancs felfüggesztése 172
10.3.3 Az NMI használata 172
10.4 A parancsok használata 172
10.4.1 A parancsok általános felépítése 172
10.4.2 Szimbólumok 173
10.4.3 Számok 173
10.4.4 Címek 173
10.4.5 Cím-tartomány 173
10.4.6 Sorszámok 173
10.4.7 Karaktersorozatok 174
10.4.8 Kifejezések 174
10.5 A SYMDEB parancsai 175
10.5.1 Az Assemble (fordítás) parancs 175
10.5.2 A BreakPoint Set (töréspont megadása) parancs 175
10.5.3 A Breakpoint Clear (töréspont törlése) parancs 176
10.5.4 A Breakpoint Disable (töréspont kikapcsolása) parancs 176
10.5.5 A Breakpoint Enable (töréspont bekapcsolása) parancs 176
10.5.6 A Breakpoint List (töréspont kiírása) parancs 176
10.5.7 A Compare (összehasonlítás) parancs 176
10.5.8 A Display ? (kifejezés értékének kiírása) parancs 177
10.5.9 A Dump ASCII parancs 177
10.5.10 A Dump Bytes parancs 177
10.5.11 A Dump Words parancs 177
10.5.12 A Dump Doublewords parancs 177
10.5.13 A Dump Short Reals parancs 177

10.5.14 A Dump Long Reals parancs 177
10.5.15 A Dump Ten-Byte Reals parancs 178
10.5.16 A Dump parancs 178
10.5.17 Az Enter parancs 178
10.5.18 Az eXamine Symbol Map parancs 178
10.5.19 A Fill parancs 178
10.5.20 A Go parancs 178
10.5.21 A Help ? parancs 179
10.5.22 A Hex parancs 180
10.5.23 Az Input parancs 180
10.5.24 A Load parancs 180
10.5.25 A Move parancs 181
10.5.26 A Name parancs 181
10.5.27 Az Open Map XO parancs 181
10.5.28 Az Output parancs 181
10.5.29 A PTrace parancs 181
10.5.30 A Quit parancs 181
10.5.31 A Redirection <, >, = parancs 181
10.5.32 A Registers parancs 182
10.5.33 A Search parancs 182
10.5.34 A Set Source Mode parancs 182
10.5.35 A Trace parancs 183
10.5.36 Az Unassemble parancs 183
10.5.37 A Write parancs 183
10.6 Hibaüzenetek 183
10.7 A SYMDEB kompatibilis fordítók 184

11. **KERESZTHIVATKOZÁSI-LISTA KÉSZÍTŐ PROGRAM (CREF)** 185
11.1 Bvezetés 185
11.2 A CREF HASZNÁLATA 185
11.3 A kereszthivatkozási-lista 186
11.4 Hibaüzenetek 186

12. **HIBAJELZÉSEK** 187
12.1 Az assembler hibaüzenetei 187
assembler hibák 187
I/O hibák 192
12.2 A szerkesztő hibaüzenetei 192

1. Bevezetés

A macro-assembler a 8086/80186/80286 mikroprocesszor-család utasításainak fordítására készült. A 8086, 8088, 8087, 80186, 80286 és 80287 (későbbi változatai a 80386) mikroprocesszorok utasításait tudja feldolgozni.

A 8086 és a 8088 mikroprocesszorok funkcionális szempontból lényegében azonosak. A köztük lévő szerkezeti eltérés a működési sebességben nyilvánul meg: a 8086 gyorsabb.

A 8087 és a 80287 az ún. lebegőpontos koprocesszorok.

A 80186 és a 80286 mikroprocesszorok a 8086 utasításkészletét bővítik további utasításokkal.

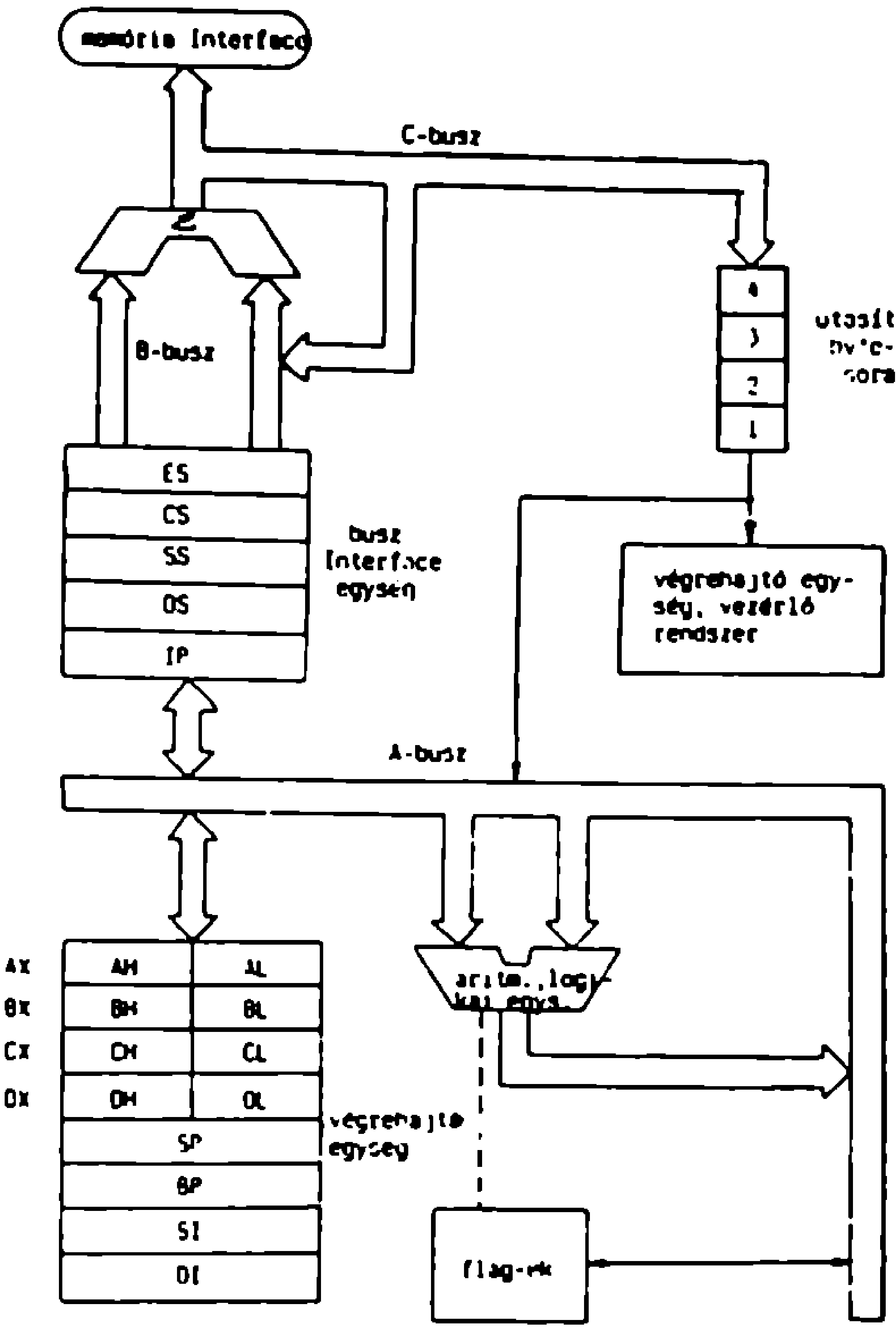
A macro-assembler alapértelmezésben csak a 8086/88 és a 8087 utasításait ismeri fel. Ha a forrásprogram a 80186, 80286 vagy 80287 utasításokat tartalmaz, akkor a programban meg kell adni a megfelelő utasításkészletet beállító direktívá(ka)t.

A könyvben használt jelölések

[ ]	A szögletes zárójelek között lévő elemet nem kötelező megadni.
	Azt jelzi, hogy az előtte lévő elem ismételhető.
	Azt jelzi, hogy az előtte lévő elem ismételhető, és az egyes ismételt elemeket vesszővel kell elválasztani egymástól.
VASTAG NAGYBETŰ	Utasítás vagy paraméter neve, amelyet pontosan a megadott formában kell használni.
dőltbetű	Olyan azonosító, amelyet a programozó maga határozhat meg.
|	Az egymástól függőleges vonallal elválasztott elemek közül pontosan egyet kell megadni. Ha ezeket a lehetőségeket szögletes zárójelpárba tettük, akkor az el is maradhat. Például [+ | –] azt jelenti, hogy ezen a helyen semmi, a + vagy a – jel szerepel.

2. A mikroprocesszor-család architektúrája

2.1 A 8088 CPU blokkdiagramja


memória interface
C-busz
Σ
B-busz
4
3
2
1
utasítás byte-ok sora
ES
CS
SS
DS
IP
busz interface egység
végrehajtó egység, vezérlő rendszer
A-busz
AX
BX
CX
DX
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
végrehajtó egység
aritm., logikai egys.
flag-ek

A processzor és a memória közötti adatforgalom a memória interface-en keresztül bonyolódik le. A processzorhoz érkező adatok az utasítás byte-ok sorába kerülnek. A végrehajtó egység vezérlő rendszere kiveszi az utasításokat ebből a sorból és a végrehajtó egységnek továbbítja, amely értelmezi és végrehajtja azokat. Ameddig ez a folyamat zajlik, a busz interface egység megpróbálja újra feltölteni az utasítás byte-ok sorát. Ez lényegesen megrövidíti az utasítások végrehajtási idejét, mivel - a legtöbb mikroprocesszortól eltérően - a következő végrehajtandó utasítás már a processzorba került a memóriából, nem kell várni a beolvasásra.

A végrehajtó egység aritmetikai/logikai egységet is tartalmaz, amely a számításokat és az összehasonlításokat végzi. A flag-ek (jelző bitek) az eredmény állapotát jelzik (pl. nulla, negatív).

2.2 Regiszterek

Programozói szempontból a regiszterek a mikroprocesszor legfontosabb alkotóelemei, mivel az assembly nyelvű programban állandóan hivatkozni kell rájuk. Mindegyik regiszter 16 bit hosszúságú.

Négy **általános célú regiszter** van: AX, BX, CX és DX. Ezek felső vagy alsó byte-jára külön is lehet hivatkozni. A felső (magasabb helyiértékű) byte-ot az AH, BH, CH, DH az alsó (alacsonyabb helyiértékű) byte-ot az AL, BL, CL, DL regiszternév jelöli.

Az SI és DI regiszterek a karaktersorozatok indexregiszterei. SI-ben a **küldő mező indexét**, DI-ben a **fogadó mező indexét** szokás tárolni. SI és DI általános regiszterként is használható.

Az SP veremmutató regiszter lehetővé teszi hardware verem implementálását a memóriában. Az SP mindig a verem tetején lévő elemre mutat. A BP bázis-regiszterrel a veremben lévő adatokat érhetjük el. A BP regisztert általános regiszterként is lehet használni.

Négy szegmensregiszter használatos a memória címzéséhez: CS, DS, SS és ES. A mikroprocesszor mindig hivatkozik egy vagy több szegmensregiszterre, amikor egy utasítást végrehajt.

Az IP regiszter utasításmutató regiszter, amely a következő végrehajtandó utasítás címét tartalmazza.

A flag-ek regisztere egybites információkat tartalmaz a mikroprocesszor aktuális állapotáról. Ezek leírása a II. fejezet 5. pontjában található.

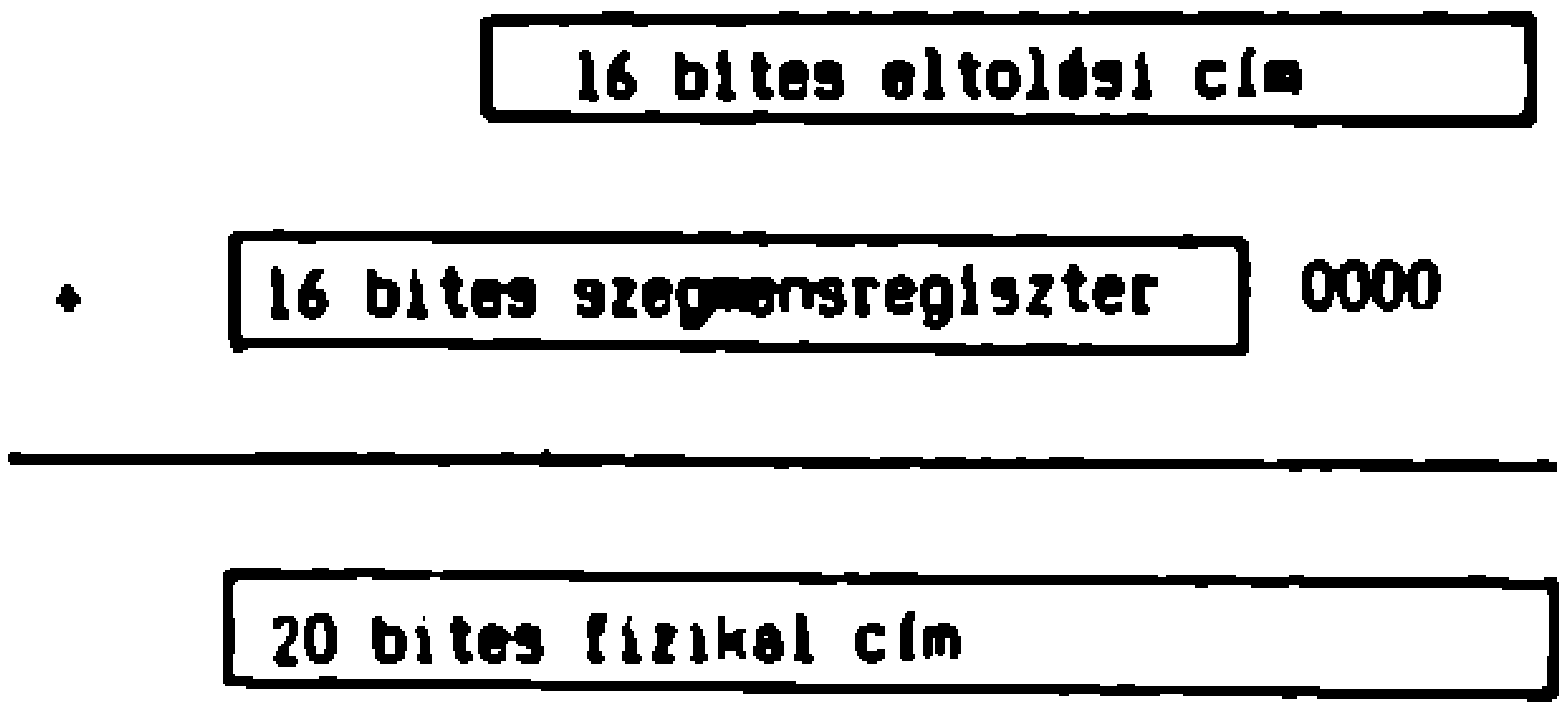
2.3 Tárcímzés szegmentálással

A 8086 mikroprocesszor család 16-bites címekkel hivatkozik a központi memóriára. Egy 16-bites címmel legfeljebb 64 Kbyte-ot (=65536 byte-ot) lehet megcímezni. A memória szegmensekre bontásával és a szegmensek címzésével a tárkapacitás 1 Mbyte-ig (1048576 byte) bővíthető.

1 Mbyte címzéséhez 20-bites címszóra van szükség. A memóriaszegmensek legfeljebb 64K méretűek lehetnek, ezért a szegmensen belüli címzés 16 biten megoldható.

Minden szegmensnek paragrafushatáron kell kezdődnie, ami azt jelenti, hogy valamennyi szegmens kezdőcíme 16-tal osztható. A szegmensek átlapolhatók, vagyis egy adott byte több szegmens kezdőcímén keresztül is elérhető lehet.

A mikroprocesszor úgy számítja ki a memória tényleges címét, hogy először kiválasztja azt a szegmensregisztert, amelyben az objektumot tartalmazó szegmens címe található. Ezt az értéket 4 bit-tel balra lépteti (megszorozza 16-tal), így megkapja a szegmens fizikai kezdőcímét, majd ehhez hozzáadja az objektum szegmens-relatív, más néven eltolási címét. Az eredmény az objektum teljes, 20-bites fizikai címe. A következő ábra ezt a műveletet mutatja.



A szegmensregiszterek a következő szegmenstípusok kezdőcímét tartalmazzák:

CS: CODE szegmens, amelyben a végrehajtandó utasítások találhatók

DS: DATA szegmens, amelyben adatok illetve sztringkezelő utasítások input adatmezői vannak

SS: STACK szegmens, amelyet a veremkezelő műveletek végrehajtásakor használunk; az elemekhez való hozzáférést a BP bázisregiszter biztosítja

ES: EXTRA szegmens, amely másodlagos adatterületet és sztringkezelő műveletek output adatmezőit foglalhatja magába.

Egy utasítás címének kiszámításakor a processzor a CS regiszterben lévő szegmenscímet és az IP utasításszámlálóban lévő eltolási címet használja fel.

Az eddigiekből következik, hogy egyidejűleg legfeljebb 4x64 Kbyte = 256 Kbyte tárterületet lehet elérni.

Ha a program kevés adattal dolgozik, nincs szükség EXTRA szegmensre. Ilyenkor ES tartalma a DS tartalmával egyenlő, tehát az EXTRA szegmens azonos az adatszegmenssel.

Minthogy a szegmensregiszterek írhatók és olvashatók, a szegmensek elhelyezését dinamikusan lehet változtatni.

2.4 Adatcímzési módok

Az adatokra következőképpen hivatkozhatunk:

- regiszterrel: pl. BX vagy CH
- konstanssal vagy literállal: pl. 5 vagy 'C'
- közvetlen memóriacímmel: pl. ADAT
- indirekt címzéssel: pl. [BX]
- bázis-relatív címzéssel: pl. [BX + 5]
- bázis-relatív indexelt címzéssel: pl. [BX + SI + 5]

Regiszterrel való hivatkozáskor az adat az adott 8 vagy 16 bites regiszterben van.

A **konstanssal** vagy **literállal** való hivatkozás azt jelenti, hogy az adatot beírjuk az utasításba.

Közvetlen memóriacímmel egy azonosító által kijelölt memóriaterület tartalmára hivatkozunk.

Indirekt címzés esetén az adott terület tartalma címként értelmeződik, és az e címen lévő memóriatartalomra hivatkozunk.

Ha **bázis-relatív címzést** adunk meg, akkor a bázisregiszter tartalmának és az eltolásnak az összege adja azt a tárcímet, amelyen a hivatkozott adat van.

Bázis-relatív indexelt címzés esetén az adat fizikai címe a bázisregiszter tartalmának, az indexregiszter tartalmának és az eltolásnak az összege.

Az indirekt, bázis-relatív és bázis-relatív indexelt címzésekben a BX, BP, SI és DI regiszterekre lehet hivatkozni.

Konstans vagy literál csak küldő mezőként szerepelhet az utasításokban. A többi címzési módot akár küldő, akár fogadó mezők esetében lehet használni az utasítás típusától függően.

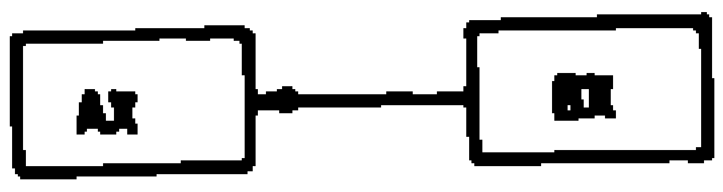
Ha egy tárbeli operandusra hivatkozunk, akkor a processzor azt feltételezi, hogy az adat a DATA szegmensben van és a címkiszámításkor a DS szegmensregiszter tartalmát veszi figyelembe. *Egyetlen kivétel ez alól az az eset, amikor a BP regisztert bázisregiszterként használjuk*: ekkor az SS regiszteren keresztül a verem-szegmensre hivatkozunk. Ezért ha pl. az elérendő adat az EXTRA szegmensben van, akkor ezt külön jelölni kell: pl. ES:[BX].

Példák:

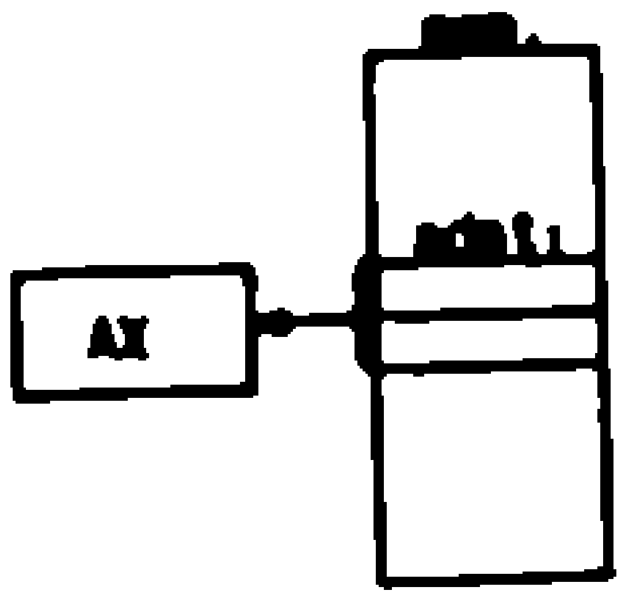
A következő példákban olyan műveleteket ábrázolunk, amelyek az AX regiszterbe egy egyszavas adatot töltenek be.

művelet / címzési mód / működés

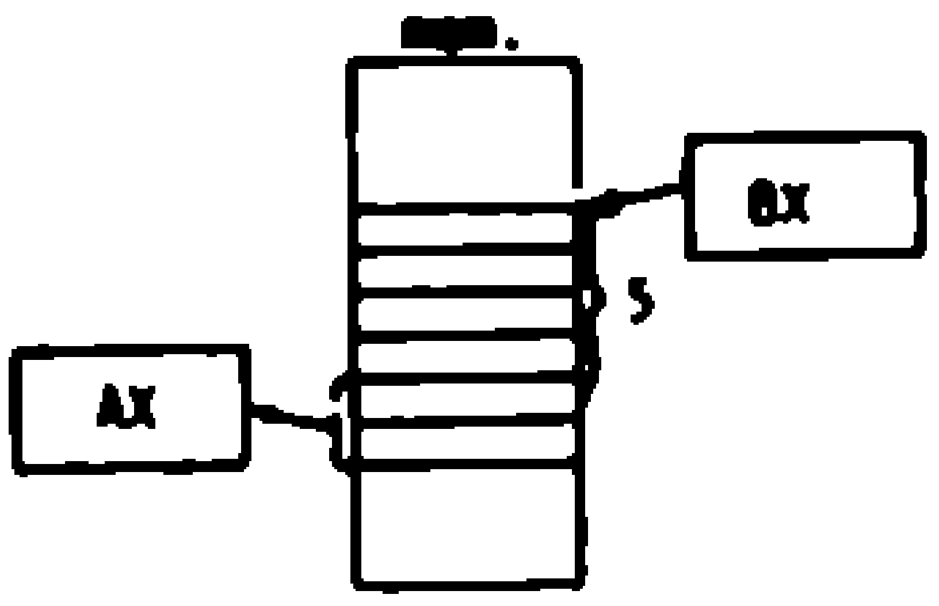
MOV AX,BX
regiszter



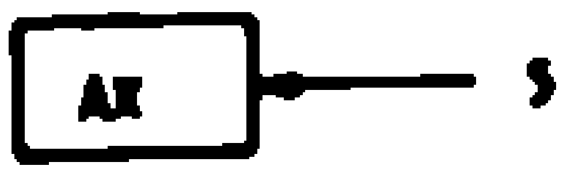
MOV AX,ADAT
közvetlen memória



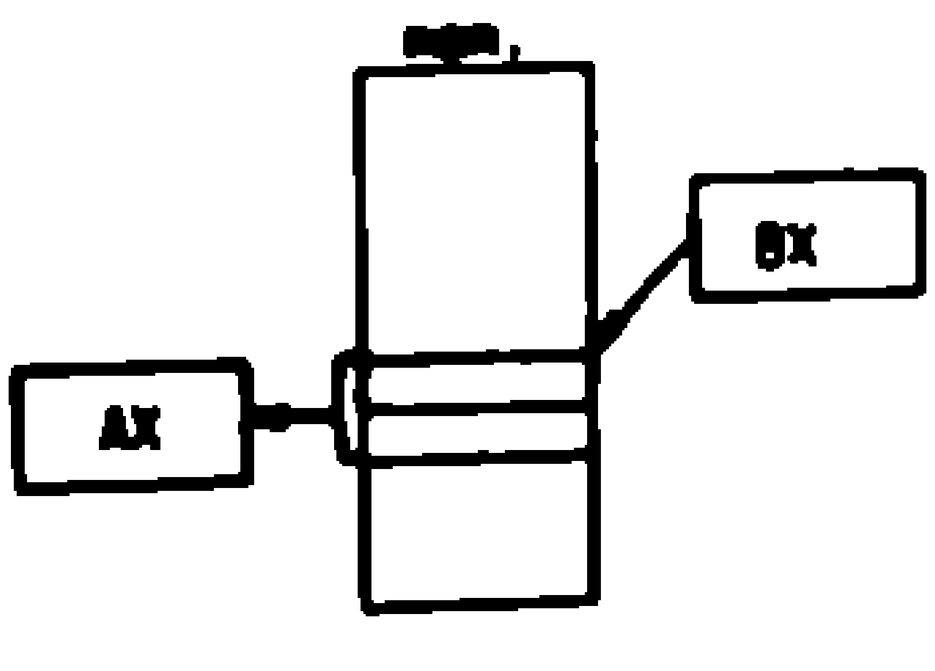
MOV AX,[BX + 5]
bázis-relatív



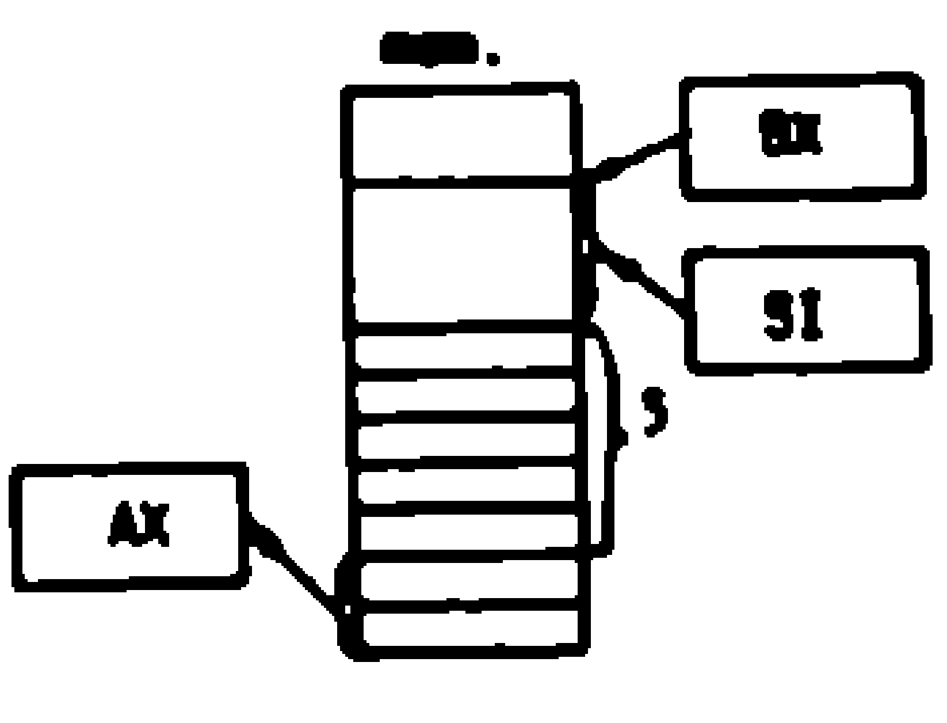
MOV AX,5
konstans



MOV AX,[BX]
indirekt



MOV AX,[BX + SI + 5]
bázis-relatív indexelt



2.5 Az utasításkészlet fő csoportjai

Az utasításkészlet azon elemi műveletek összessége, amelyeket a mikroprocesszor végre tud hajtani. Ezeket funkció szerint 6 fő csoportba soroljuk:

- adatátviteli utasítások
- aritmetikai utasítások
- logikai utasítások
- karaktersorozat-kezelő utasítások
- vezérlésátadó utasítások
- processzorvezérlő utasítások.

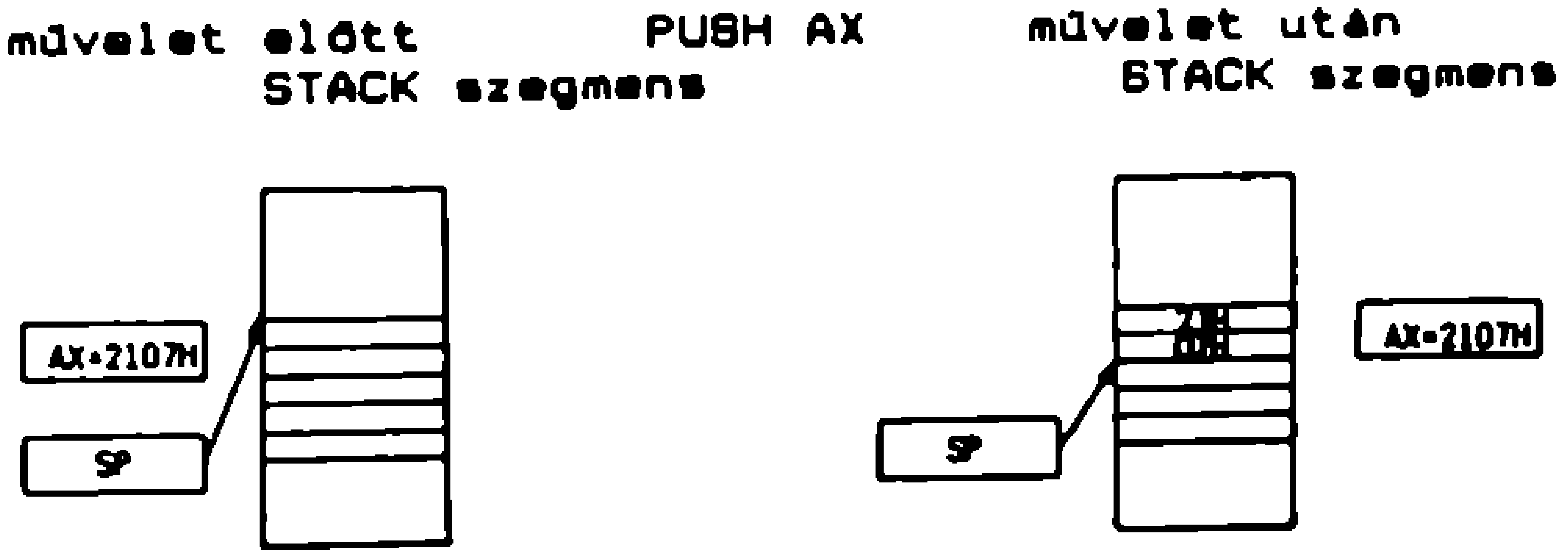
Ebben a fejezetben az utasítások működésének lényegét írjuk le. Pontos szintaktikájuk a III. fejezetben található.

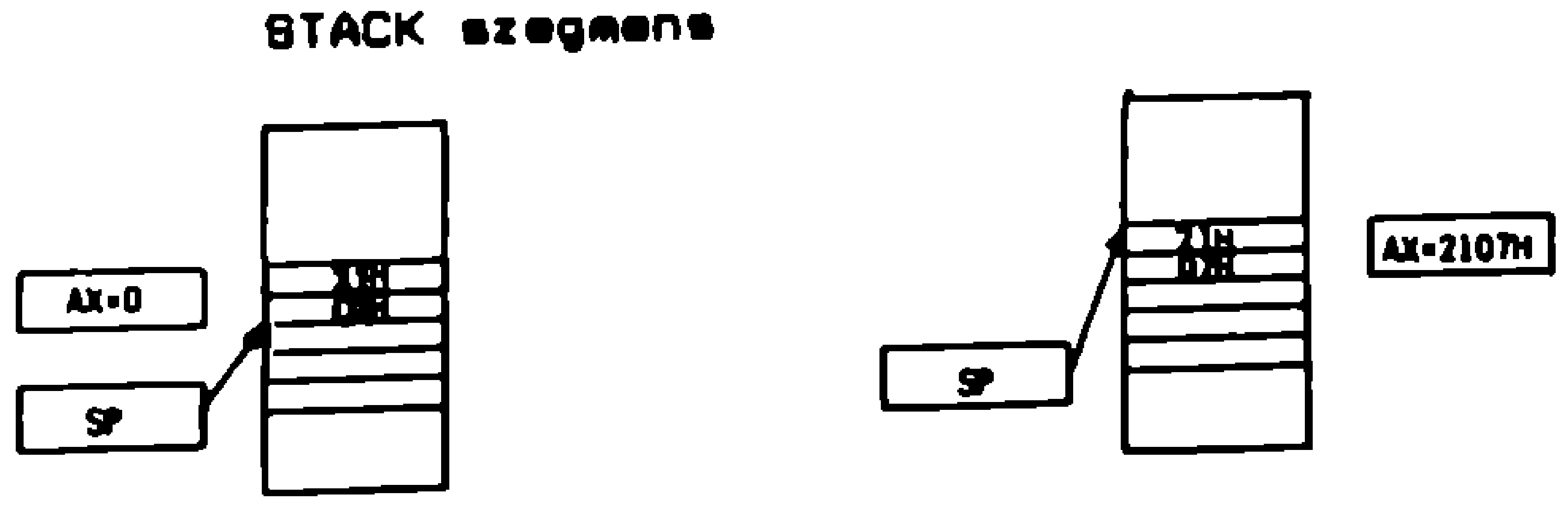
2.5.1 Adatátviteli utasítások

Az adatátviteli utasítások a következők:

- operandus feltöltése: MOV
- veremkezelő műveletek: POP, PUSH
- két operandus tartalmának felcserélése: XCHG
- input/output műveletek: IN, OUT
- konvertálás: XLAT

A veremkezelő műveletek közül a PUSH adatot helyez el a veremben, a POP pedig adatot vesz ki a veremből. *Mindig szavas adatműveletet végeznek.* Működésüket a következő ábrák szemléltetik:


művelet előtt
STACK szegmens
PUSH AX
művelet után
STACK szegmens
AX=2107H
SP
21H
07H
AX=2107H
SP


művelet előtt
STACK szegmens
POP AX
művelet után
AX=0
SP
21H
07H
AX=2107H
SP
21H
07H

A veremből az utoljára betett adatot lehet először kivenni. A **veremkezelő műveleteket** általában a program adatainak mentésére, szubrutinhíváskor és a megszakítási technika kezelésekor szokás használni.

Az **input/output műveletek** i/o portokon (kapukon) keresztül biztosítják a mikroprocesszor és a külső készülékek (terminál, nyomtató, lemez) közötti kommunikációt. Az i/o portok byte-os vagy szavas adatok átvitelére alkalmasak. Az adatokat az AX regiszterből kell elküldeni (OUT) ill. az AX regiszterbe kerülnek (IN). Ha az adat csak egy byte-os, akkor elegendő az AL regiszter használata. Az i/o port címét a DX regiszter tartalmazza, vagy az utasításban konstansként/literálként adjuk meg. Az utóbbi esetben csak az első 256 i/o port érhető el.

A **konvertálást végző XLAT utasítás** úgy működik, hogy a BX regiszter tartalmához hozzáadja az AL regiszter tartalmát, majd az eredményt címként értelmezve az e címen lévő byte-ot betölti az AL regiszterbe. Általában úgy használják, hogy BX tartalma egy konverter-tábla kezdőcíme.

2.5.2 Aritmetikai utasítások

Az aritmetikai utasítások a következők:

- összeadás: ADD
- inkrementálás: INC
- kivonás: SUB
- dekrementálás: DEC
- kettes komplemens képzése: NEG
- összehasonlítás: CMP
- szorzás: MUL
- osztás: DIV

Ezeket a műveleteket 1 byte vagy 1 szó hosszúságú operandusokon lehet végezni. Minden aritmetikai utasítás az elvégzett művelet eredményétől függően beállítja a flag-regiszter megfelelő bitjeit. A feltételes vezérlésátadó utasításokkal a flag-ek tartalma lekérdezhető és programelágaztatást lehet végrehajtani.

2.5.3 Logikai utasítások

A logikai utasítások a következők:

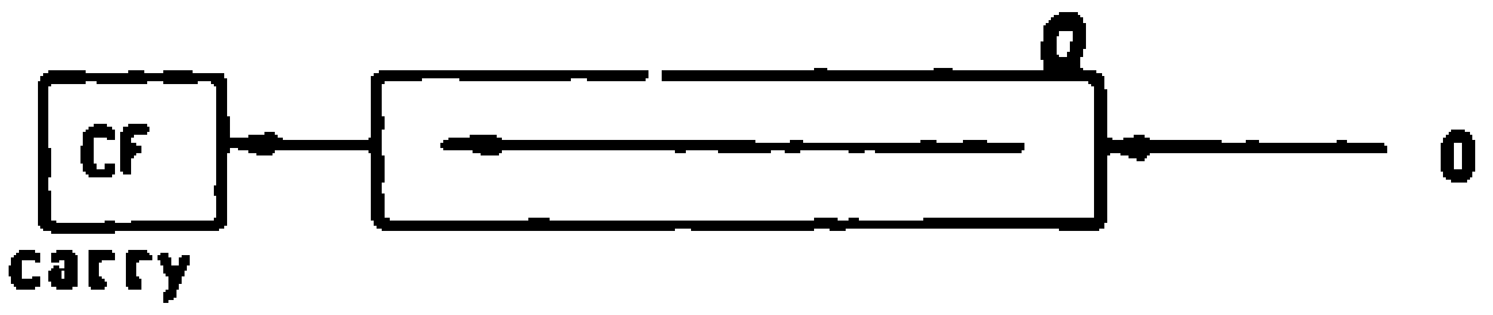
- tagadás: NOT
- "és" művelet: AND
- "vagy" művelet: OR
- "kizáró vagy" művelet: XOR
- byte vagy szó bitjeinek tesztelése: TEST
- léptetés balra: SHL, SAL
- léptetés jobbra: SHR, SAR
- ciklikus léptetés balra: ROL, RCL
- ciklikus léptetés jobbra: ROR, RCR

A logikai utasítások nem numerikus értékeknek, hanem bitsorozatoknak tekintik az operandusokat.

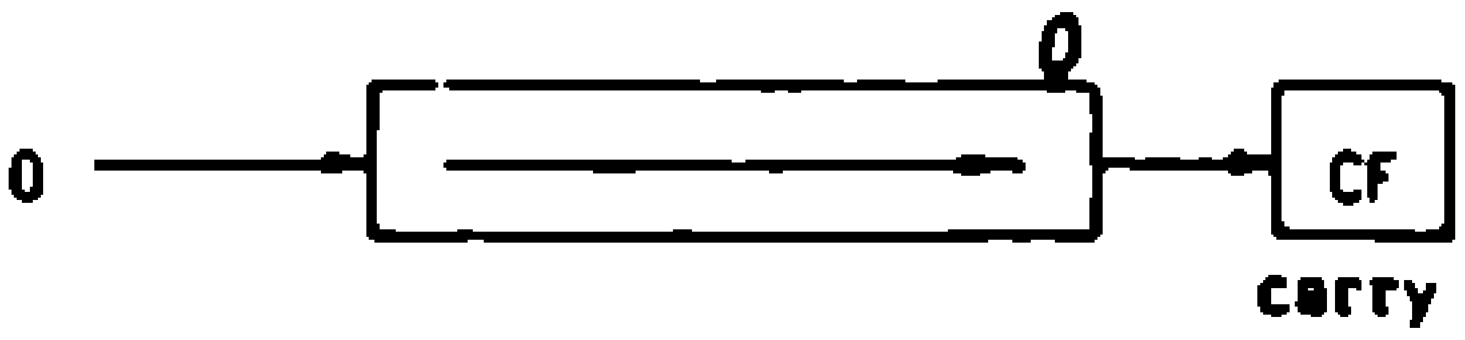
A NOT, AND, OR, XOR és TEST logikai utasítások mindig törlik a carry és az overflow flag-et és a zero flag-et 1-re állítják, ha a művelet eredménye 0. (A TEST is logikai "és" műveletet hajt végre, akárcsak az AND, de az operandusok változatlanok maradnak.)

Ha a bitsorozat numerikus értéket ábrázol, akkor a balra és jobbra léptető utasításokat 2 hatványaival való műveletvégzésre is használhatjuk. A balra léptetéssel szorzást, a jobbra léptetéssel osztást végezhetünk. Működésüket a következő ábra mutatja:

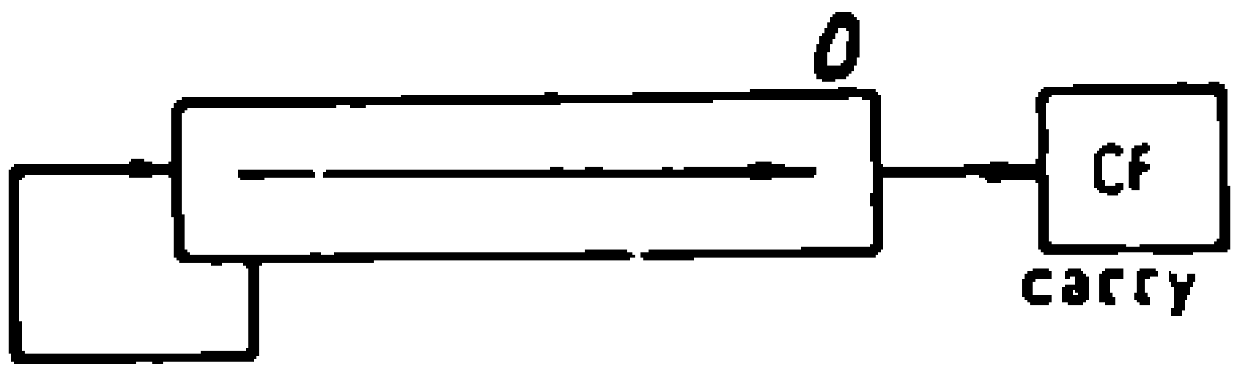
balra léptetés
(SHL)



jobbra léptetés
(SHR)



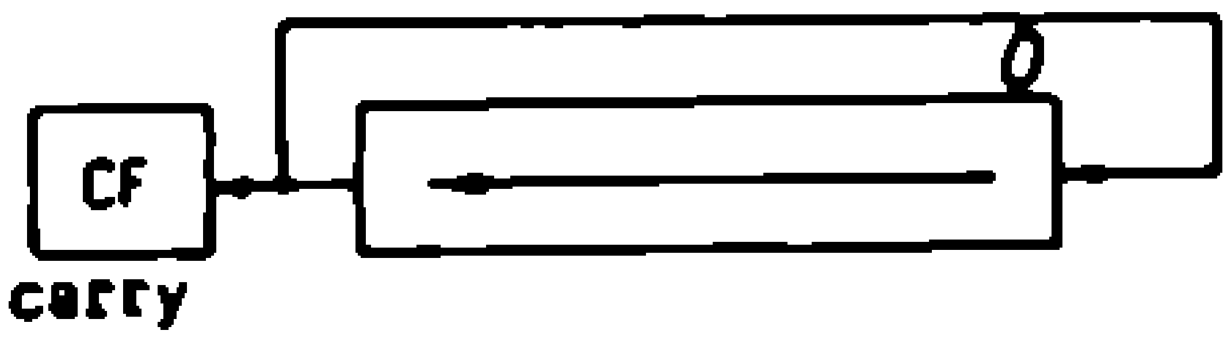
Ha a léptetendő operandus előjeles numerikus érték (2-es komplemens formában ábrázolva), akkor a jobbra léptetést, azaz az osztást aritmetikai léptetésként is végezhetjük (SAR utasítás). Ekkor a bal szélső (legmagasabb helyiértékű) bit változatlan marad. Az aritmetikai jobbra léptetés működési ábrája:



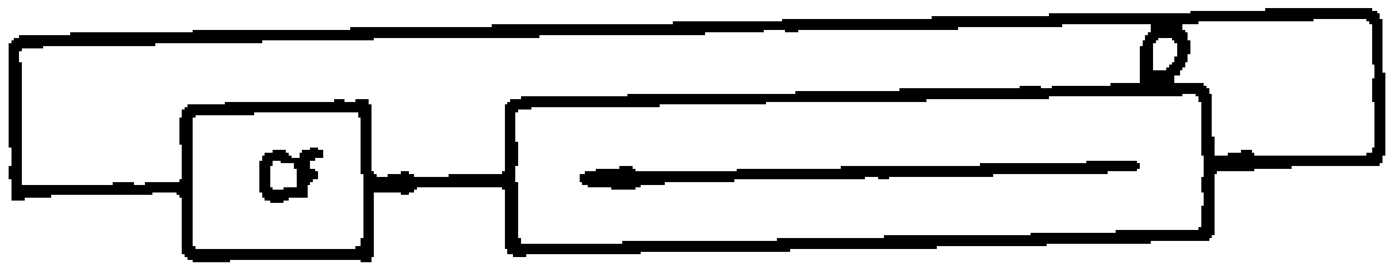
A ciklikus léptető utasítások esetében a kilépő bitek nem "vesznek el", hanem 0 helyett ezekkel töltődnek fel a szabaddá váló bitek.

Működésük:

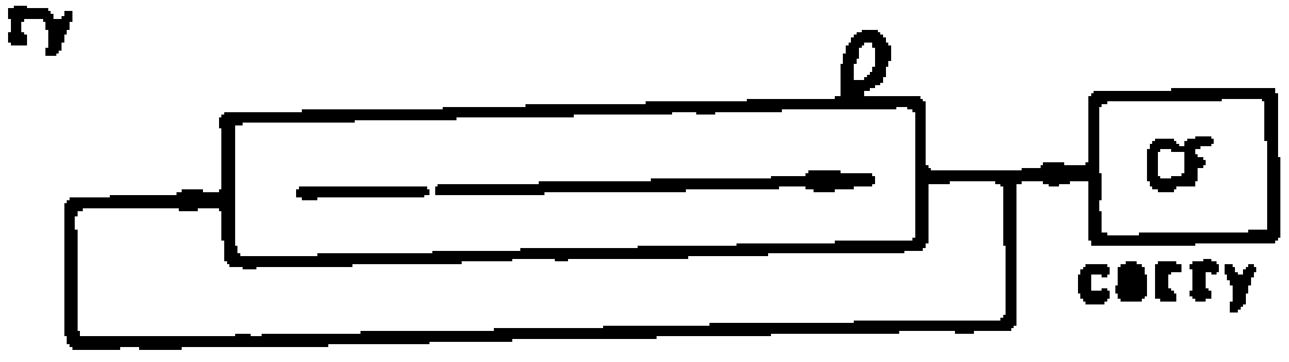
ciklikus balra léptetés
(ROL)



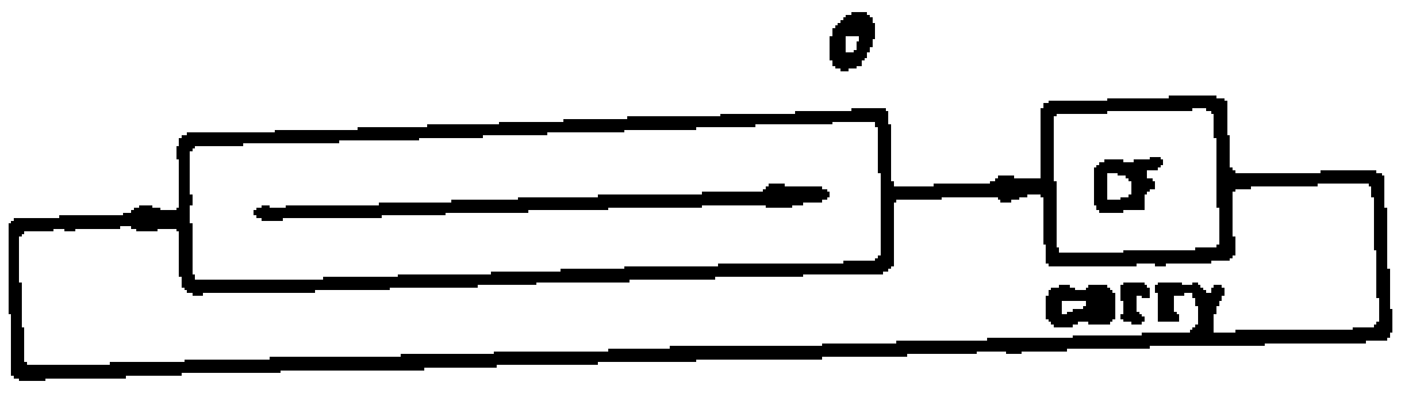
ciklikus balra léptetés a carry flag-en keresztül
(RCL)



ciklikus jobbra léptetés
(ROR)



ciklikus jobbra léptetés a carry flag-en keresztül
(RCR)



A logikai műveletek operandusai 1 byte vagy 1 szó hosszúságúak lehetnek.

2.5.4 Karaktersorozat-kezelő utasítások

A karaktersorozatok kezelését könnyítő utasítások a következők:

- karaktersorozat másolása: MOVS
- két karaktersorozat összehasonlítása: CMPS
- adott byte vagy szó keresése egy karaktersorozatban: SCAS
- karaktersorozat egy byte-jának vagy egy szavának betöltése az AL vagy az AX regiszterbe: LODS
- az AL vagy az AX regiszter tartalmából karaktersorozat előállítása: STOS.

Mindegyik karaktersorozat-kezelő utasítás 1 byte vagy 1 szó hosszúságú operandusokkal működik.

A **karaktersorozat-kezelő utasítások** operandusait nem az utasításban adjuk meg, hanem megállapodás szerinti helyeken. Eszerint a küldő mező elemeit az SI regiszterrel címezzük, a fogadó mező elemeit a DI regiszterrel. A **fogadó mezőnek** mindig az EXTRA szegmensben kell lennie. A **küldő mező** alapértelmezés szerint a DATA szegmensben helyezkedik el. Ha másik szegmensben lévő küldő mezőre akarunk hivatkozni, akkor explicit módon meg kell adni a mezőt tartalmazó szegmens szegmensregiszterét. A küldő és a fogadó mező azonos szegmensben is lehet. Ilyenkor a DS és az ES szegmensregiszterek tartalmának azonosnak kell lennie.

A karaktersorozat-kezelő műveleteket a kisebb memóriacímtől a nagyobb felé haladva, vagy fordított irányban, a nagyobb címtől a kisebb felé haladva lehet végrehajtani. **A feldolgozás iránya az irányt jelző flag értékétől függ**: ha a flag értéke 0, akkor a címek növekvő sorrendjében történik a feldolgozás, ha a flag értéke 1, akkor a címek csökkenő sorrendjében.

A karaktersorozat-kezelő művelet végrehajtása előtt a DS és az SI regiszter a küldő mező kezdőcímét, az ES és a DI regiszter a fogadó mező kezdőcímét határozza meg. A karaktersorozat-kezelő utasítás automatikusan inkrementálja vagy dekrementálja SI és DI értékét a feldolgozás irányától függően.

A karaktersorozat egy byte-jának vagy szavának az AL illetve az AX regiszterbe való töltése kivételével a string-kezelő utasítások ismételhetők (REP, REPE, REPNE). Ilyenkor a karaktersorozat hosszát a CX regiszterben kell elhelyezni. A művelet végrehajtása során CX tartalma automatikusan csökken.

Az AL vagy AX tartalmának tárolásakor az SI tartalma változatlan marad, csak DI módosul, mert nincs forrás-karaktersorozat. A művelet végrehajtása előtt CX az előállítandó karaktersorozat hosszát tartalmazza, a művelet végrehajtása után CX tartalma nulla lesz.

Egy karaktersorozat egy byte-jának vagy egy szavának betöltésekor az AL illetve az AX regiszterbe a DI tartalma változatlan marad, csak SI módosul, mert nincs fogadó karaktersorozat.

Ha egy adott byte-os vagy szavas értéket akarunk keresni egy karaktersorozatban, akkor a keresett értéket AL-ben vagy AX-ben helyezzük el. Az SCAS utasítás kivonja AL vagy AX tartalmából az ES és DI regiszterrel címzett byte-ot vagy szót. A műveletvégzés során csak az összehasonlítás eredményét jelző flag, és a DI regiszter tartalma módosul. Az összehasonlítást vagy akkor hagyjuk abba, amikor egyezést találunk (REPNE), vagy akkor, amikor eltérést találunk (REPE). Az ismétlési tényezőt a keresés előtt CX tartalmazza. A keresés végrehajtása után CX tartalma azoknak a byte-oknak vagy szavaknak a száma, amelyek a string kezdetétől haladva a keresett byte-ot vagy szót megelőzték.

A két karaktersorozatot összehasonlító utasítás rendre kivonja a forrás-mező byte-jaiból vagy szavaiból a fogadó mező byte-jait vagy szavait. A művelet hatására az operandusok nem változnak, csak az összehasonlítás eredményét jelző flag, valamint SI és DI tartalma módosul. Az összehasonlítást vagy akkor hagyjuk abba, amikor egyezést találunk (REPNE), vagy akkor, amikor eltérést találunk (REPE). Az összehasonlítás ismétlési tényezőjét a műveletvégzés előtt CX tartalmazza. Műveletvégzés után CX tartalma azoknak a byte-oknak vagy szavaknak a száma, amelyek nem kerültek összehasonlításra.

Az SI illetve DI regisztert módosító karaktersorozat-kezelő műveletek végrehajtása után SI illetve DI az utoljára feldolgozott byte vagy szó után soron következő címre mutat, figyelembe véve a feldolgozás haladási irányát.

2.5.5 Vezérlésátadó utasítások

Mint már említettük, a processzor byte-onként olvassa be a memóriából a végrehajtandó utasításokat. A következő végrehajtandó utasítás mindig a CS regiszter tartalma által meghatározott aktuális kódszegmensnek az utasítás-számláló regiszter (IP) által meghatározott relatív címén van. A program végrehajtása általában az utasítások felírási sorrendjében történik. Minden utasítás végrehajtása után az IP regiszter tartalma megfelelő mértékben növekszik..

A vezérlésátadó utasítások lehetővé teszik, eltérjünk a szekvenciális programfutástól. Ennek két alapvető módja van:

- az IP regiszter tartalmának megváltoztatásával ugyanabban a kódszegmensben, de más relatív címen folytatódik a program futása - ezt **szegmensen belüli vezérlésátadásnak** nevezzük.
- a CS és az IP regiszter tartalmának egyidejű megváltoztatásával egy másik kódszegmensben levő címen folytatódik a program futása - ez a **szegmensek közötti vezérlésátadás**.

A következő vezérlésátadó utasításokat lehet használni:

- feltétel nélküli ugrás: JMP
- feltételes ugrások: JZ, JNZ,...
- feltétel nélküli ciklus: LOOP
- feltételes ciklus: LOOPE, LOOPNE,...
- szubrutinhívás: CALL
- szubrutinból való visszatérés: RET
- megszakítás: INT
- megszakításból való visszatérés: IRET

Feltétel nélküli ugrást szegmensen belül vagy szegmensek között lehet végrehajtani. Szegmensen belüli vezérlésátadás esetében az utasítás operandusaként megadott cím lehet:

- direkt
    - 16 bittel címezhető, a szegmensen belül tetszőleges helyen levő címke;
    - 8 bittel címezhető, a vezérlésátadó utasítástól 128 byte-nál kisebb távolságra lévő címke (SHORT típusú ugrás)
- indirekt
    - egy regiszter, amely címet tartalmaz
    - egy memóriacím, amely címet tartalmaz.

Szegmensek közötti vezérlésátadás esetében az utasítás operandusaként megadott cím lehet:

- direkt:	egy másik szegmensben deklarált címkeé
- indirekt:	egy memóriacím, amelyen lévő szó tartalma az IP regiszterbe, az ezt követő szó tartalma a CS regiszterbe kerül.

A feltételes ugró utasítások hatására akkor következik be vezérlésátadás, ha az utasításkóddal meghatározott feltétel teljesül. A feltételek a jelzőbitek (flag-ek) tartalmára vonatkoznak. A flag-ek álttalában az aritmetikai és logikai utasítások eredményét jelzik. A zero, sign, overflow és parity flag közvetlenül tesztelhető. A feltételes vezérlésátadó utasításokat csak szegmensen belüli, direkt címzési módú és 128 byte-nál kevesebb távolságú ugrásokra lehet használni.

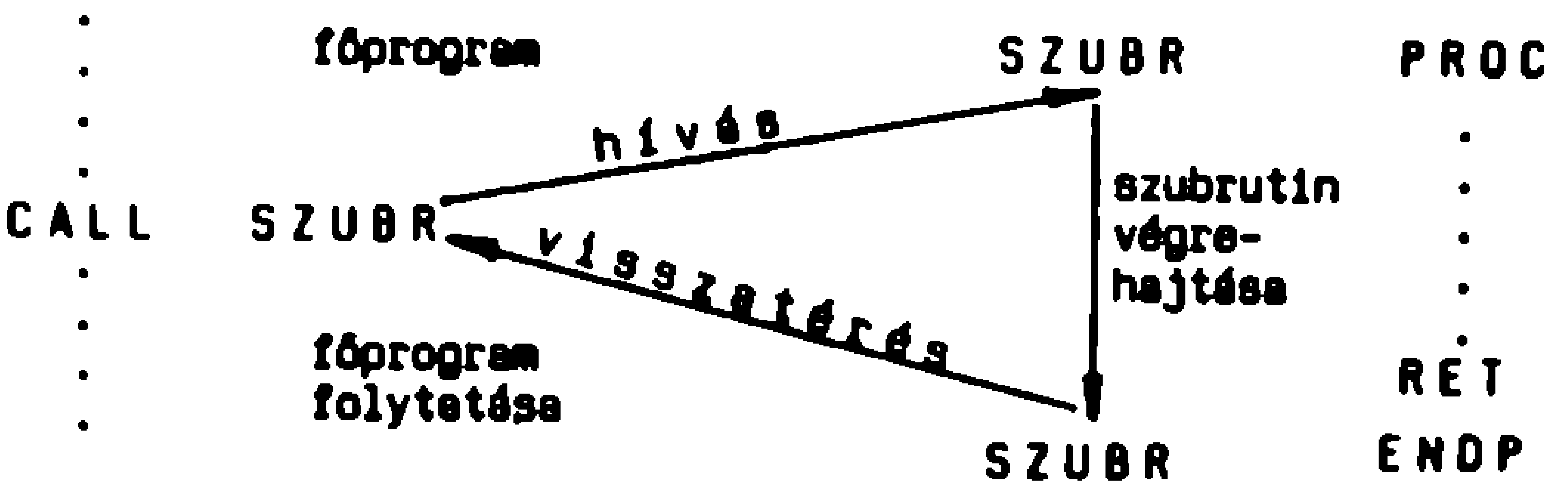
Az előbb említett megkötés (szegmensen belüli, direkt címzési mód, 128 byte-nál kisebb távolság) a **ciklusszervező utasításokra** is érvényes. A ciklusszervező utasítások lehetővé teszik, hogy egy utasítássorozat végrehajtását megismételjük.

Az ismétlendő sorozat előtt a CX regiszterben kell tárolni az ismétlési tényezőt, majd címkével kell ellátni a ciklusmag első utasítását. A ciklusmag minden végrehajtása után a CX tartalma automatikusan csökken. Háromféle ciklust lehet szervezni:

- akkor fejeződik be az ismétlés, ha CX = 0 (LOOP);
- akkor fejeződik be az ismétlés, ha CX = 0 vagy a zero flag értéke 0 (LOOPE, LOOPZ);
- akkor fejeződik be az ismétlés, ha CX = 0 vagy a zero flag értéke 1 (LOOPNE, LOOPNZ).

A ciklus befejezése után a program végrehajtása a ciklusszervező utasítást követő utasítással folytatódik.

A szubrutinok olyan utasítássorozatok, amelyek a program tetszőleges pontjánál hívhatók. A CALL utasítás hatására a vezérlés a szubrutin első utasítására kerül. A szubrutin utolsó utasítása a RET, amely visszaadja a vezérlést a hívó program CALL utasítását követő első utasítására. Ezt a folyamatot a következő ábrán láthatjuk:



A főprogram és a szubrutin közötti paraméterátadást a vermen keresztül szokás lebonyolítani.

Szegmensen belüli és más szegmensben definiált szubrutin egyaránt hívható. A CALL utasítás operandusára ugyanazok a szabályok vonatkoznak, mint a feltétel nélküli ugró utasítás (JMP) operandusára.

A CALL végrahajtásakor a visszatérési cím automatikusan a verem tetejére kerül. A RET végrehajtásakor a processzor kiveszi a visszatérési címet a veremből. *A verem használata lehetővé teszi a szubrutinok egymásba ágyazását és biztosítja az utolsó hívás helyére való visszatérést.*

Az I/O eszközökről a mikroprocesszorhoz érkező jeleket **megszakításoknak** (interrupt) nevezzük. A megszakításokat a mikroprocesszor **interrupt-mechanizmusa szolgálja ki**, amely két alapvető lehetőséget nyújt:

- letilthatja, hogy a mikroprocesszor érzékelje a megszakításokat, ezt az interrupt flag 0-ra való állításával lehet elérni, kivéve a nem maszkolható megszakításokat (NMI), amelyek érzékelése és kezelése az interrupt flag tartalmától független;
- a megszakítás fogadása után menti a mikroprocesszor aktuális állapotát, így a megszakítás kiszolgálása után folytatható az abbahagyott tevékenység (az aktuális állapot mentése úgy történik, hogy a flag-regiszter tartalma a verembe kerül, az interrupt és a trap flag törlődik, majd a CS és az IP regiszter tartalma is a verembe kerül.

A processzor aktuális állapotának mentése után a megszakítási mechanizmus a beérkezett megszakításoknak megfelelő kiszolgáló rutinra adja a vezérlést. Ha ez a rutin lefutott (azaz a megszakítást kezelő rutin egy IRET utasítást hajt végre), akkor az IRET hatására a processzor kiveszi a veremből az IP-t, a CS-t, és a flag-ek regiszterét, és a vezérlés arra a programra kerül, amelyet a processzor a megszakítás pillanatában hajtott végre.

A megfelelő kiszolgáló rutin kiválasztása az INT utasítás operandusa, egy 1 byte-os típuskód alapján történik. Ez azt jelenti, hogy legfeljebb 256 kiszolgáló rutin hívása lehetséges. A programban az 5-255 típuskódok adhatók meg, mivel a 0-4 típuskódot a mikroprocesszor saját magának foglalja le:

0 - a **nullával való osztás** kezelése
1 - **nyomkövetés**
2 - **nem maszkolható megszakítás**
3 - **speciális 1 byte-os megszakítások** (nincs típuskód operandusa az INT utasításnak)
4 - **túlcsordulás** kezelése (INTO utasítás)

2.5.6 Processzorvezérlő utasítások

A processzorvezérlő utasítások az átviteli, a feldolgozás irányát jelző és a megszakítás flag-et állítják be. A HLT utasítás felfüggeszti a processzor működését. Ezt akkor használjuk, ha egy megszakításra való várakozás ideje alatt semmilyen tevékenységet nem akarunk végrehajtani.

A flag-eket állító processzorvezérlő utasítások a következők:

- a carry flag törlése: CLC
- a direction flag törlése: CLD
- az interrupt flag törlése: CLT
- a carry flag ellekezőjére állítása: CMC
- a carry flag 1-re állítása: STC
- a direction flag 1-re állítása: STD
- az interrupt flag 1-re állítása: STI.

3. Adattípusok

3.1 Karakterkészlet

Minden assembly nyelvű program **utasításokból** és **megjegyzésekből** áll. Az utasítások kialakításához a következő **karakterkészlet** elemeit lehet használni:

ABCDEFGHIJKLMNOPQRSTUVWXYZ (angol nagybetűk)

abcdefghijklmnopqrstuvwxyz (angol kisbetűk)

0123456789 (számjegyek)

?@ $:.[]()< >{}

+ -/*&%!_ ‡\ = # ;,'" (speciális jelek és az aláhúzás)

3.2 Numerikus konstansok

3.2.1 Egész számok

Alakjuk:

számjegy ...[típusjel]

Az egész szám bináris, oktális, decimális vagy hexadecimális számjegy(ek)ből és opcionális típusjelből áll. Ha a típusjelet nem adjuk meg, akkor a fordító az aktuális alapfeltételezést használja, ami a program indításakor a decimális típus. *A típus alapértelmezését a .RADIX direktívával változtathatjuk meg.* A típusjelet a programban akár nagy-, akár kisbetűvel meg lehet adni. A .RADIX direktíva részletes leírása a IV. fejezet 4.8 pontjában található.

Az egész számban megadható jegyek maximális száma attól függ, hogy milyen utasításban vagy direktívában használjuk.

A következő táblázat azt mutatja be, hogy melyik típus milyen számjegyeket tartalmazhat:

Típusjel	Típus	Számjegyek
B	bináris	0 1
Q	oktális	0 1 2 3 4 5 6 7
O	oktális	0 1 2 3 4 5 6 7
D	decimális	0 1 2 3 4 5 6 7 8 9
H	hexadecimális	0 1 2 3 4 5 6 7 8 9 A B C D E F

A hexadecimális számoknak mindig decimális számjeggyel (0-9) kell kezdődniük. A hexadecimális jegyek (A-F) nagy- és kisbetűvel is írhatók.

Példák:

101B 173Q 4FH 69D 69 0FH 9ABH

3.2.2 Valós számok

Alakjuk:

számjegy... . *számjegy*... [E[+ | -]*számjegy*...]

A valós számok egész részből, törtrészből és esetleg kitevőből állnak. A számjegyek decimális jegyek. A kitevő használata nem kötelező.

Valós számokat csak a DD, DQ és DT direktívában lehet használni. A szám jegyeinek maximális száma és a kitevő maximális értéke az adott direktívától függ.

A DD, DQ és DT adatdeklaráló direktívák pontos leírása a IV. fejezet 2.1 pontjában található.

Példák:

98.76 9.876E1 9876.E-2

3.2.3 Kódolt valós számok

Alakjuk:

számjegy ... **R**

A **kódolt valós szám** egy 8-, 16- vagy 20-jegyű hexadecimális szám. A kódolt valós szám előjelmezőből, kitevőből és mantisszából áll. Ezek az értékek bitmezőkként tárolódnak a memóriában. A bitmezők pontos mérete és jelentése a szám bitjeinek számától függ. A számnak decimális jeggyel kell kezdődnie (0-9).

Kódolt valós számokat csak a DD, DQ és DT direktívában lehet használni. A jegyek száma vezető nullával való megadás esetén pontosan 9, 17 vagy 21 kell hogy legyen.

Példák:

3F000000R
3F00000000000000R

3.2.4 Tömörített decimális számok

Alakjuk:

[+ | -]*számjegy*...

A **tömörített decimális számok** decimális egészek, amelyek tömörített decimális formában tárolódnak. Tömörített decimális tárolás esetén az előjel-byte-ot 9 érték-byte követi. Minden érték-byte 2 decimális jegyet tartalmaz. Az előjel-byte legmagasabb helyiértékű bitje 0, ha a szám pozitív, és 1, ha a szám negatív. Mindig 10 byte-on ábrázolódnak.

A tömörített decimális számok olyan alakúak, mint a decimális egészek, attól eltekintve, hogy előjelesek is lehetnek és csak a DT direktívában használhatóak. A tömörített decimális számok legfeljebb 18-jegyűek lehetnek.

Példák:

```
97540      ; kódolva: 00000000000000097540
-97540     ; kódolva: 80000000000000097540
```

3.3 Karakter- és karaktersorozat-konstansok

Alakjuk:

'karakter...'
"karakter..."

A **karakterkonstans egyetlen ASCII karakterből áll.** A karaktersorozat-konstans két vagy több ASCII karakter kombinációja. Mindkét konstanst aposztrofok (') vagy idézőjelek (") közé kell tenni.

Ha a határoló jel a konstans belsejében is szerepel, akkor ott kettőzni kell.

Példák:

'x' "x"
'cdef'
"paraméterhiba"
'ilyen "kulcs" nem létezik' de "ilyen 'kulcs' nem létezik"

3.4 Azonosítók

Alakjuk:

karakter...

Az **azonosító betűk, számjegyek és speciális karakterek kombinációja**, amelyek címkeként, változóként vagy szimbólumként lehet használni.

Az azonosítóra a következő formai szabályok vonatkoznak:

1. Betűvel vagy a _ ? $ @ speciális karakterek valamelyikével kell kezdődnie.
2. Az azonosítóban vegyesen szerepelhetnek kis- és nagybetűk. A kisbetűk nagybetűkké konvertálódnak, ha nem használjuk a /ML vagy /MX opciót a fordító elindításakor.
3. Az azonosító tetszőleges hosszúságú lehet, de a macro-assembler csak az első 31 karaktert veszi figyelembe.

Példák:

HIBA2
_alprogram
Feldolgozás

3.5. Operandusok

A macro-assembly program utasításaiban és direktíváiban különböző operandusokat használhatunk. Az operandus konstans, címke, változó vagy egyéb szimbólum lehet, amely az utasítás vagy direktíva által végzendő műveletben értéket, regisztert vagy memóriaterületet jelöl.

Az utasításokban és a direktívákban használható operandus-típusok:

- konstans
- direkt címmel elérhető adat
- relatív címmel elérhető adat
- számláló
- regiszter
- bázis-relatív
- indexelt
- bázis-relatív indexelt
- struktúra
- rekord
- rekordmező

3.5.1 Konstans operandusok

Alakjuk:

szám | *karaktersorozat* | *kifejezés*

A konstans a többi operandustól eltérően nem memóriacímet jelent, hanem konkrét értéket, amelyet egy számmal, egy karaktersorozat-konstanssal, egy szimbólummal vagy egy kifejezéssel adhatunk meg. A kifejezések felépítésére vonatkozó tudnivalók a II. fejezet 6. pontjában találhatók.

Példák:

szám: 123
karaktersorozat-konstans: 'abcd'
szimbólum: alma
kifejezés: 5*7

3.5.2 Direkt címmel elérhető operandusok

Alakjuk:

szegmens:eltolás

A direkt címmel elérhető operandus a tár egy vagy több bytejának abszolút címét adja meg. A "szegmens" szegmensregiszternév (CS, DS, SS, ES), szegmensnév vagy csoport neve lehet. Az eltolás egy olyan érték, amely 0 és 65535 közé esik és amelyet egész számmal, abszolút szimbólummal vagy kifejezéssel lehet megadni.

Példák:

DS:26
adatok:26
csoport:elem

3.5.3 Relatív címmel elérhető operandusok

Alakjuk:

szimbólum

A relatív címmel elérhető operandus olyan szimbólum, amely egy utasítás vagy adat memóriacímét jelenti. A direkt címmel elérhető operandustól eltérően annak a szegmensnek vagy csoportnak a kezdetéhez relatív, amelyben az adott szimbólumot definiáltuk. Konkrét értéket csak a programszerkesztés folyamán kap.

Példa:

cím

3.5.4 Számláló operandus

A számlálóra a $ jellel hivatkozhatunk. A számláló a fordítás során az aktuális szegmens aktuális helyét jelenti. Ez egy olyan utasításcím, amely az aktuális szegmens kezdetéhez relatív. Az eltolást a fordító számolja ki oly módon, hogy a szegmens minden utasításnak lefordítása után annyival növeli a számláló értékét, ahány byte-ot generált. Az operandus végleges értékét csak a szerkesztés folyamán kapja meg.

3.5.5 Regiszter-operandusok

Alakjuk:

regiszternév

A regiszternév egy processzor-regiszter neve, amely a következők valamelyike lehet:

AX AH AL BX BH BL
CX CH CL DX DH DL
CS DS SS ES SP BP
DI SI

A regiszterekre való hivatkozáskor nagy- és kisbetűk tetszőleges kombinációja megengedett.

Az AX, BX, CX és DX 16-bites általános célú regiszterek, amelyeket aritmetikai vagy logikai műveletek operandusaiként használhatunk. Az általános regiszterek alsó és felső 8 bitjére külön-külön is lehet hivatkozni. A megfelelő regiszterek alsó 8 bitjére való hivatkozást AL, BL, CL és DL, a felső 8 bitjére való hivatkozást AH, BH, CH és DH jelöli.

CS, DS, SS és ES szegmensregisztert azonosít. CS a kódszegmens, DS az adatszegmens, SS a veremszegmens, ES a különleges szegmens címét tartalmazza. Minden utasítás és adat címe a szegmensregiszterek valamelyikében lévő címhez relatív.

Az SP regiszter 16-bites és a verem-mutatót (stack pointer) tartalmazza, amely a verem tetejének aktuális címe. Ez a cím az SS regiszterben lévő szegmenscímhez relatív. A vermet használó utasítások automatikusan módosítják az SP regiszter tartalmát.

BX, BP, DI és SI 16-bites bázis- és indexregiszterek. Ezeket az általános regisztereket a program adataihoz való hozzáféréskor használhatjuk.

A 16-bites flag-regiszternek nincs azonosítója. A regiszter tartalma a LAHF, SAHF, PUSHF és POPF utasítások használatával hozzáférhető. A flag-regiszter kilenc egybites flag-et tartalmaz, amelyek helye és jelentése a következő:

Pozíció	**Jelentés**
0	Carry (átvitel): aritmetikai műveleteknél a legmagasabb helyiértékű biten keletkezett átvitel.
2	Parity (paritás): értéke 1, ha egy 8-bites adattal végzett művelet eredményeképpen páros számú 1-es bit keletkezett, egyébként 0.
4	Auxiliary carry (segéd-átvitel): 8-bites adattal végzett művelet esetén a 3. és 4. bit közt keletkezett átvitel.
5	Trap (nyomkövetés): ha értéke 1, a processzor nyomkövető üzemmódban működik (azaz minden utasítás után egy INT 1 megszakítást generál).
6	Zero (nulla eredmény): értéke 1, ha egy aritmetikai, vagy logikai művelet eredménye 0.
7	Sign (előjel): aritmetikai művelet végrehajtásakor keletkezett eredmény legmagasabb helyiértékű bitje.
9	Interrupt-enable (megszakítás-engedélyezés): ha értéke 1, a megszakítások engedélyezettek.
10	Direction (irány): ha 1 az értéke, akkor a feldolgozó a karaktersorozat (sztring) végénél kezdődik (az SI és DI indexregiszterek tartalma automatikusan csökken), egyébként a feldolgozás a sztring elején kezdődik (SI, DI pedig értéke növekszik).
11	Overflow (túlcsordulás): értéke aritmetikai művelet végrehajtása után a legmagasabb és az azt megelőző helyiértékű biten keletkezett átvitellel végzett kizáró "vagy" művelet eredménye.

3.5.6 Bázis-relatív operandusok

Alakjuk:

eltolás **[BX]**
eltolás **[BP]**

A szögletes zárójel itt nem jelölés, leírása kötelező!

A bázis-relatív operandusok a BX vagy BP bázisregiszterek valamelyikéhez relatív memóriacímeket jelentenek. Az eltolás közvetlen vagy közvetett memória-operandus lehet, amelynek értéke egy szám vagy egy memóriacím. Ha nem adjuk meg az eltolást, akkor 0 az alapértelmezés. A bázis-relatív operandus tényleges címe az adott regiszter tartalmának és az eltolásnak az összege. A BX bázisregiszter használata esetén a kiszámított cím a DS regiszter által kijelölt szegmenshez relatív. Ha a BP regisztert használjuk, akkor a bázis-relatív operandus címe az SS regiszter által kijelölt szegmenshez relatív.

A bázis-relatív operandusokat az eddig leírtaktól eltérő formában is megadhatjuk, pl. [BP].eltolás, vagy [BX + -eltolás], de a címkiszámítási algoritmust ez nem befolyásolja.

3.5.7 Indexelt operandusok

Alakjuk:

eltolás **[DI]**
eltolás **[SI]**

A szögletes zárójel itt nem jelölés, kiírása kötelező!

Az indexelt operandus a DI vagy SI indexregiszterek valamelyikéhez relatív címet jelent. Az eltolás közvetlen vagy közvetett memória-operandus lehet, amelynek értéke egy szám vagy egy memóriacím. Ha nem adjuk meg az eltolást, akkor 0 az alapértelmezés.

Az indexelt operandus tényleges címe az adott regiszter tartalmának és az eltolás értékének az összege. Az így kiszámított cím mindig a DS regiszter által kijelölt szegmenshez relatív.

Az indexelt operandusokat az eddig leírtaktól eltérő formában is megadhatjuk, pl. [DI].eltolás, vagy [SI + -eltolás], de ez nem befolyásolja a címkiszámítási algoritmust.

3.5.8 Bázis-relatív indexelt operandusok

Alakjuk:

eltolás **[BX][DI]**
eltolás **[BX][SI]**
eltolás **[BP][DI]**
eltolás **[BP][SI]**

A szögletes zárójel itt nem jelölés, kiírása kötelező!

A bázis-relatív indexelt operandusok a bázis- és indexregiszter kombinációjához relatív memóriacímeket jelentenek. Az eltolás közvetlen vagy közvetett memória-operandus lehet, amelynek értéke egy szám vagy egy memóriacím. Ha nem adjuk meg az eltolást, akkor 0 az alapértelmezés.

A bázis-relatív indexelt operandus tényleges címe az adott regiszterek tartalmának és az eltolás értékének az összege. Ha a BP regisztert használjuk, akkor a kiszámított cím az SS regiszter által kijelölt szegmenshez relatív, egyéb esetben viszont a DS regiszter által kijelölt szegmenshez.

A bázis-relatív indexelt operandusokat az eddig leírtaktól eltérő formában is meg lehet adni, pl. [eltolás][BP][DI], vagy [BX + SI + -eltolás], de ez a címkiszámítási algoritmust nem befolyásolja.

3.5.9 Struktúra-operandusok

Alakjuk:

változó.mező

A struktúra-operandus egy struktúra valamely elemének memóriacímét jelenti. A *változó* vagy egy struktúra azonosítója, vagy egy memória-operandus, amely egy struktúra címét tartalmazza. A *mező* a struktúra valamely elemének az azonosítója. A struktúrák deklarációját a 5. fejezet 2.4 pontja tartalmazza.

A struktúra-operandus tényleges címe a *változó* és a *mező* relatív címének az összege. A cím ahhoz a szegmenshez vagy csoporthoz relatív, amelyben a változót definiáltuk. A struktúra-operandust gyakran használjuk veremben lévő adatok eléréséhez. Ennek egyik módja az, hogy az aktuális veremcímet a BP regiszterbe tesszük és a verem elemeire a következőképpen hivatkozunk:

[BP].elem

Így a verem minden elemét tetszőleges formában elérhetjük.

3.5.10 Rekord-operandusok

Alakjuk:

rekordnév < [*érték*],,, >

A rekord-operandus egy rekordtípus adataira hivatkozik. Az operandusok kifejezésekben is szerepelhetnek. A *rekordnév* egy olyan rekordtípus azonosítója, amelyet a forrás-file-ban definiáltunk. Az opcionális *érték* a rekord valamely mezőjének a tartalma. Több érték megadása esetén az adatokat vesszővel kell elválasztani egymástól. Ha egy mező nincs kitöltve, a fordító annak alapértelmezésével fog dolgozni. A csúcsos zárójelek megadása akkor is kötelező, ha benne egyetlen érték sem szerepel.

A rekordok megadási módját a 5. fejezet 2.5 pontja tartalmazza.

3.5.11 Rekordmező-operandusok

Alakjuk:

rekordmezőazonosító

A rekordmező-operandus a mezőnek a megfelelő rekordban való helyét jelenti. Az operandus értéke a mező legalacsonyabb helyiértékű bitjének a pozíciója. Konstans operandusként használható. A rekordmező azonosítójának egy előzőleg már definiált mező azonosítójának kell lennie.

3.6 Az operandusokból előállítható kifejezések

A kifejezések operandusokból és műveleti jelekből állnak. A kifejezésben felírt műveletek eredményeképpen egy értéket vagy egy memóriacímet kapunk attól függően, hogy milyen típusú operandusokat és műveleti jeleket használtunk. A műveleti jelek típusai:

- aritmetikai
- léptető
- összehasonlító
- logikai
- attributum-kezelő

3.6.1 Aritmetikai műveletek

A műveletek jelölése:

kifejezés1 * *kifejezés2*
kifejezés1 / *kifejezés2*
kifejezés1 MOD *kifejezés2*
kifejezés1 + *kifejezés2*
kifejezés1 - *kifejezés2*
+*kifejezés*
-*kifejezés*

A műveleti jelek jelentése:

két operandusú műveletek:

*	szorzás
/	osztás
MOD	osztás utáni maradék (modulus)
+	összeadás
-	kivonás

egy operandusú műveletek:

+	pozitív
-	negatív

Aritmetikai műveletet - az összeadás és a kivonás kivételével - csak egész számokkal lehet végezni. Az összeadás műveletével egy relokálható memória-operandushoz hozzáadhatunk egy egész számot. Lehetőség van egy relokálható operandusból egész szám kivonására, vagy két relokálható operandus különbségének kiszámítására is. Ez utóbbi csak akkor lehetséges, ha a relokálható operandusok azonos szegmensben helyezkednek el.

Példák:

3 * 2 = 6
3 / 2 = 1
3 MOD 2 = 1

3 + 2 = 5
3 - 2 = 1
3 - +2 = 1
3 - -2 = 5
a + 2
a - 2
a - b

3.6.2 Léptető műveletek

Felírási módjuk:

kifejezés SHR *lépésszám*
kifejezés SHL *lépésszám*

Az SHR művelettel annyi bittel léptetjük jobbra az adott kifejezést, amennyit lépésszámként megadtunk. SHL-lel ugyanígy balra lehet léptetni. A kilépő bitek elvesznek. Ha a lépésszám > = 16, akkor az eredmény 0 lesz.

Példák:

10101010B SHR 5 = 00000101B
10101010B SHL 5 = 01000000B

3.6.3 Logikai műveletek

Felírási módjuk:

kifejezés1 EQ *kifejezés2*
kifejezés1 NE *kifejezés2*
kifejezés1 LT *kifejezés2*
kifejezés1 LE *kifejezés2*
kifejezés1 GT *kifejezés2*
kifejezés1 GE *kifejezés2*
NOT *kifejezés*
kifejezés1 AND *kifejezés2*
kifejezés1 OR *kifejezés2*
kifejezés1 XOR *kifejezés2*

Az összehasonlító műveletekkel két olyan kifejezést hasonlíthatunk össze, amelyek konkrét értékkel rendelkeznek. Az összehasonlító művelet eredménye igaz logikai érték (0FFFFH), ha a feltétel teljesül, ellenkező esetben hamis logikai érték (0000H). Az összehasonlító műveleteket a fordítás vezérlésére használjuk feltételes direktívákban és feltételes utasításokban. Az összehasonlító műveletek jelentése:

EQ	kifejezés1 = kifejezés2
NE	kifejezés1 < > kifejezés2
LT	kifejezés1 < kifejezés2
LE	kifejezés1 < = kifejezés2

GT	kifejezés1 > kifejezés2
GE	kifejezés1 > = kifejezés2

Példák:

20 EQ 10	hamis
20 NE 10	igaz
20 LT 10	hamis
20 LE 10	hamis
20 GT 10	igaz
20 GE 10	igaz

A logikai műveletekben szereplő kifejezéseknek konkrét értékkel kell rendelkezniük. A kifejezések tetszőleges típusúak. Hosszúságuk attól függ, milyen utasításban használjuk a logikai műveletet. Az egyes műveleti jelek jelentése:

NOT	negáció
AND	logikai "és"
OR	logikai "vagy"
XOR	logikai "kizáró vagy"

A logikai műveletek bitenként hajtódnak végre.

Példák:

NOT 00000001B = 11111110B
11110000B AND 00001111B = 00000000B
11110000B OR 00001111B = 11111111B
11110000B XOR 00011111B = 11101111B

3.6.4 Indexelés

Felírási módja:

kifejezés1[kifejezés2]

A felírási módban ismét nem tudtuk jelölésrendszerünket alkalmazni. "Kifejezés1" opcionális, "kifejezés2"-t viszont kötelező megadni és *szögletes zárójelpár közé tenni, mivel itt a szögletes zárójelpár az indexelés műveleti jele.*

Az indexelést általában egy tömb vagy egy karaktersorozat elemeire való hivatkozáskor használjuk. Az indexelés művelete megegyezik az összeadás műveletével, kivéve, ha a "kifejezés1"et nem adjuk meg. Ha "kifejezés1"-et elhagyjuk, akkor az indexelési művelet eredményeképpen az egész típusú 0 értéket kapjuk.

"Kifejezés1" egész szám, abszolút szimbólum vagy relokálható operandus lehet. Ha "kifejezés1" nem szerepel, "kifejezés2"-nek egész számnak vagy abszolút szimbólumnak kell lennie.

Egyéb esetben "kifejezés2" is egész szám, abszolút szimbólum vagy relokálható operandus lehet.

Példák:

tömb[6]
string[2]

3.6.5 A PTR művelet

Felírási módja:

típus **PTR** *kifejezés*

A PTR műveletet előrehivatkozáskor használjuk azért, hogy így explicit módon adjuk meg a hivatkozott objektum méretét vagy távolságát. Előrehivatkozás alatt azt értjük, hogy azt a szimbólumot, amelyre hivatkozunk, csak később definiáljuk a programban. Ha előrehivatkozáskor nem alkalmazzuk a PTR műveletet, a fordító az objektum típusára vonatkozó alapfeltételezést használja, és ez hibajelzéshez vezethet, ha az alapfeltételezés nem felel meg az objektum későbbi definíciójának.

A PTR művelettel a kifejezésben megadott változó vagy címke típusát határozhatjuk meg. A "típus" a következő kulcsszavak vagy az azokat jelölő értékek valamelyike lehet:

BYTE	1
WORD	2
DWORD	4
QWORD	8
TBYTE	10
NEAR	0FFFFH
FAR	0FFFEH

A "*kifejezés*" tetszőleges operandus lehet. A BYTE, WORD és DWORD típusokat csak memória-operandusokra lehet megadni. A NEAR és FAR típus csak címkékre használható.

Példa:

NEAR PTR *címke1*
FAR PTR *címke2*

NEAR típusú címkére csak abban a szegmensben lehet hivatkozni, amelyben azt definiáltuk. FAR típusú címkére a program bármely szegmensében hivatkozhatunk. (Az adatok és címkék típusára vonatkozóan a IV. fejezet 2.1 és 2.2 pontjában található bővebb információ.)

3.6.6 A címkiszámítás viszonyítási alapjának megváltoztatása

(jele a ":" karakter)

Felírási módjai:

szegmensregiszter : kifejezés
szegmensnév : kifejezés
csoportnév : kifejezés

Ha ilyen műveletet adunk meg, akkor a fordító a címke vagy változó címének kiszámításához az alapértelmezés helyett az adott szegmensregiszter, szegmensnév vagy csoportnév kezdetét használja. Ha szegmensnevet vagy csoportnevet adunk meg, akkor ezeknek olyan azonosítóknak kell lenniük, amelyeket előzőleg egy ASSUME direktívával egy szegmensregiszterhez rendeltünk és SEGMENT vagy GROUP direktívával definiáltunk. A "kifejezés" abszolút szimbólum vagy relokálható operandus lehet. Szegmensregiszterként a CS, DS, SS és ES regiszterek valamelyike szerepelhet.

A címkiszámítás alapértelmezése címkékre NEAR, a memóriaoperandusokra pedig - az utasítás és az operandus típusától függően - a DS, SS vagy ES regiszterhez relatív.

Példa:

ES:[BX][SI]

Az objektum címe az ES szegmensregiszterhez relatív. A relatív cím a BX bázisregiszter tartalmának és az SI indexregiszter tartalmának az összege. (Az ASSUME, SEGMENT és GROUP direktívák leírását a IV. fejezet 1. pontja tartalmazza.)

3.6.7 A SHORT művelet

Felírási módja:

SHORT *címke*

A SHORT művelet az adott címke típusát SHORT-ra állítja be. A SHORT címkét használó utasítás 1 byte-tal kevesebb helyet foglal, mint ha ugyanebben az utasításban NEAR típusú címkét adunk meg. A SHORT típusú címkét JMP (jump) utasításban lehet használni, ha a címkének az utasítástól való távolsága nem nagyobb, mint 127 byte.

Példa:

SHORT számok

3.6.8 A THIS művelet

Felírási módja:

THIS *típus*

A THIS művelet egy olyan operandust definiál, amelynek címe megegyezik a számláló aktuális értékével, típusát pedig a következő kulcsszavak valamelyike határozza meg:

NEAR
FAR
BYTE
WORD
DWORD
QWORD
TBYTE

A THIS műveletet az EQU vagy = direktívában használjuk. Hatása ugyanaz, mint a LABEL direktíva használatának.

Például

ciklus EQU THIS WORD és
ciklus LABEL WORD

egy olyan szó operandust definiál, amelynek címe a számláló aktuális értéke és "ciklus" az azonosítója. (Az EQU, = és a LABEL direktíva részletes leírása a IV. fejezet 2.3 pontjában található.)

3.6.9 A HIGH és a LOW művelet

Felírási módjuk:

HIGH *kifejezés*
LOW *kifejezés*

A HIGH művelet a kifejezés felső 8 bitjét, a LOW művelet a kifejezés alsó 8 bitjét választja le. A "kifejezés" tetszőleges érték lehet. A művelet a 8-bites macro assemblerekkel való kompatibilitása miatt szerepel, s használata ellenjavalt.

Példák:

HIGH 0F0F0H = 11110000B
LOW szó = 00001010B, ha a "szó" tartalma 0ED0AH

3.6.10 A SEG művelet

Felírási módja:

SEG *kifejezés*

A SEG művelettel azt a szegmensértéket kapjuk meg, amelyhez a "kifejezés" relatív. A "kifejezés" címke, változó, szegmensnév, csoportnév vagy egyéb szimbólum lehet.

Példa:

SEG adatok

3.6.11 Az OFFSET művelet

Felírási módja:

OFFSET *kifejezés*

Az OFFSET művelet az adott kifejezés eltolását adja vissza. A "kifejezés" címke, változó, szegmensnév vagy egyéb szimbólum lehet. A visszaadott érték azoknak a byte-oknak a száma, amelyek az objektum és az azt definiáló szegmens kezdete között helyezkednek el. Ha a műveletet szegmensnévre alkalmazzuk, akkor a visszaadott érték a szegmens kezdetének és a szegmens utoljára generált byte-jának távolsága.

Ha az OFFSET műveletet a címkiszámítás viszonyítási alapját megváltoztató művelettel együtt használjuk, akkor a kifejezésben megadott objektumnak az adott szegmens kezdetétől mért, byte-okban való távolságát kapjuk vissza.

A visszakapott érték mindig relatív, a program szerkesztésekor változhat.

Példa:

OFFSET csoport:vektor

Ennek hatására a "vektor" változónak a "csoport" szegmens kezdetétől byte-okban mért távolságát kapjuk vissza.

3.6.12 A TYPE művelet

Felírási módja:

TYPE *kifejezés*

A TYPE művelettel egy olyan értéket kapunk, amely az adott kifejezés típusára jellemző. Ha a kifejezés egy változó, akkor a visszaadott érték az operandus mérete byte-okban. Ha a kifejezés címke, akkor a TYPE művelet eredményeképpen NEAR típusú címke esetén 0FFFFH-t, FAR típusú címke esetében 0FFFEH-t kapunk. A visszakapott értéket fel lehet használni a típus meghatározására a PTR műveletben.

Példa:

TYPE vektor

3.6.13 A .TYPE művelet

Felírási módja:

.TYPE *kifejezés*

A .TYPE művelet hatására egy olyan byte-ot kapunk vissza, amely az adott kifejezés jellemzőit és hatáskörét tartalmazza. Ha a "kifejezés" nem definiált, akkor a visszakapott byte értéke nulla.

A változó attributumait a byte nulladik, első, ötödik és hetedik bitpozícióján kapjuk meg és jelentésük a következő:

bitpozíció	bit = 0	bit = 1
0	abszolút	programfüggő
1	nem inicializált	inicializált
5	definiálatlan	definiált
7	lokális hatáskörű	external hatáskörű (más modulban definiált)

Ha a hatáskört és a definiáltságot jelző bit értéke 0, akkor a kifejezés érvénytelen.

A .TYPE műveletet általában feltételes direktívákban használjuk, hogy az argumentum attritubumai szerint vezéreljük a programfordítást.

Példa:

```
a DB 54
b EQU .TYPE a
```

Hatására b értéke 34 lesz, mert az "a" nevű változó (= 54) abszolút, inicializált, definiált és lokális hatáskörű.

3.6.14 A LENGTH művelet

Felírási módja:

LENGTH *változó*

A LENGTH művelet hatására az adott változó elemeinek számát kapjuk meg. A mértékegység byte, szó, dupla szó, négy szó vagy tíz byte lehet attól függően, hogy a változót BYTE, WORD, DWORD, QWORD vagy TBYTE típusúra definiáltuk.

A visszakapott érték csak akkor lehet 1-nél nagyobb, ha a változó definíciójában a DUP operátor is szerepel. *A LENGTH művelet hatására ekkor mindig az első DUP operátort megelőző számot kapjuk, tekintet nélkül arra, hogy a változó definíciója tartalmaz-e több DUP operátort is.*

A DUP műveleti jel segítségével vektort vagy tömböt definiálunk úgy, hogy egy vagy több kezdőértéket is ismételhetünk. A következőképpen adható meg:

szám DUP(kezdőérték,,,)

A "szám" határozza meg, hogy hányszor ismétlődjön a kezdőérték. A kezdőérték lehet kérdőjel karakter (határozatlan kezdőértéket jelent), vagy egy olyan kifejezés, amelynek értéke egy egész szám, karakterkonstans vagy egy másik DUP művelet. A DUP műveletek 17 szint mélységig ágyazhatók egymásba. Ha egynél több kezdőértéket adunk meg, akkor a kezdőértékek sorozatonként ismétlődnek.

Például a

```
tömb DW 4 DUP(3 DUP(2 DUP(1,4,5)))
```

direktíva 24 szót foglal le és a lefoglalt szavak tartalma: 1,4,5,1,4,5,1,4,5,...

Ekkor a

```
LENGTH tömb
```

művelet eredménye 4 lesz, a

```
SIZE tömb
```

eredménye viszont 48.

3.6.15 A SIZE művelet

Felírási módja:

SIZE *változó*

A SIZE művelettel az adott változó számára lefoglalt terület byte-okban kifejezett méretét lehet lekérdezni.

3.6.16 A WIDTH művelet

Felírási módja:

WIDTH *rekordmező-azonosító | rekordazonosító*

A WIDTH művelettel az adott rekordmező vagy rekord bitekben mért hosszát kérdezhetjük le. Ha például a "rek" rekordot a következőképpen definiáltuk:

```
rtip RECORD a:2, b:5, c:1
rek  rtip   < >
```

akkor

```
WIDTH a    = 2
WIDTH b    = 5
WIDTH c    = 1
WIDTH rtip = 8
```

3.6.17 A MASK művelet

Felírási módja:

MASK *rekordmező-azonosító | rekordazonosító*

A MASK művelet hatására a rekord vagy a rekordban előforduló mezők bitmaszkját kaphatjuk meg. A maszkban azok a bitek lesznek 1 értékűek, amelyeknek megfelelő bitek az adott mezőhöz tartoznak. A rekord bitmaszkja 0FFH vagy 0FFFFH attól függően, hogy a rekord 8-bites vagy 16 bit hosszú.

Ha például a "rek" rekordot a következőképpen definiáltuk:

```
rtip RECORD a:2, b:5, c:1
rek  stip   < >
```

ami azt jelenti, hogy az "rtip" rekordtípus 8-bites, amelynek felépítése:

```
  a  |     b     | c |    mező
 7 6 | 5 4 3 2 1 | 0 |  bitpozició
```

és az "rtip" típusú "rek" rekordnak nincs kezdőértéke. Ekkor

```
MASK a    = 11000000B = 0C0H
```

MASK b = 00111110B = 3FH
MASK c = 00000001B = 01H
MASK rtip = 11111111B = 0FFH

3.6.18 A kifejezés értékének kiszámítása és a precedencia

A kifejezés értékének kiszámítása a műveleti jelek precedenciaszabályai és felírási sorrendje szerint történik. Elsőként a legmagasabb precedenciájú műveletek hajtódnak végre. *Az azonos precedenciájú műveletek elvégzése a felírási sorrendben, balról jobbra történik.* **A műveletek elvégzésének ezt a sorrendjét zárójelezéssel meg lehet változtatni.** A zárójelben lévő műveletek mindig előbb kerülnek végrehajtásra, mint a szomszédos műveletek.

A következő táblázat a műveleti jelek precedenciaszabályait foglalja össze. Az azonos sorban lévő műveletek azonos precedenciájúak. A legmagasabb precedenciájú műveletek az 1-es sorban lévők.

Precedencia	**művelet**
1	LEGTH, SIZE, WIDTH, MASK
2	()
3	[]
4	:
5	PTR, OFFSET, SEG, TYPE, THIS
6	HIGH, LOW
7	*, /, MOD, SHL, SHR
8	+, -
9	EQ, NE, LT, LE, GT, GE
10	NOT
11	AND
12	OR, XOR
13	SHORT, .TYPE

3.6.19 A memória-operandusok típusának ellenőrzése

A makro-assembler szigorú szintaktikus ellenőrzést végez az utasításokon, beleértve azokat az operandusokat, amelyek memóriaterületekre hivatkoznak. Ez azt jelenti, hogy az implicit adattípusú relokálható operandusokkal megadott utasításokban az operandusoknak vagy azonos típusúaknak kell lenniük, vagy explicit módon, a PTR művelettel azonos típusúakká kell tenni azokat. Ha ezt elmulasztjuk, a fordító hibajelzést ad.

4. A mikroprocesszor-család assembly utasításai

Ebben a fejezetben összefoglaljuk a mikroprocesszor-család assembly utasításkészletét. Az utasítások működésének lényegét az első fejezetben már ismertettük. Részletes magyarázatok és a címzési módok leírása rendelkezésre áll már a magyar számítástechnikai szakirodalomban (pl. az LSI ATSZ kiadásában: AZ i8086 mikroprocesszor), ezért itt csak az utasítások szintaktikáját és rövid jelentését soroljuk fel.

4.1 Az utasítások leírásában használt rövidítések

Rövidítés	**Értelmezés**
akku	akkumulátor: AX vagy AL regiszter
reg	egy byte vagy egy szó hosszúságú regiszter byte: AL, AH, BL, BH, CL, CH, DL, DH, szó: AX, BX, CX, DX, SI, DI, BP, SP
szreg	szegmensregiszter: CS, DS, SS, ES
ált	általános operandus: regiszter, memóriacím, indexelt operandus, bázis-relatív operandus vagy bázis-relatív indexelt operandus
konst	8- vagy 16-bites konstans vagy literál
mem	memória-operandus: címke, változó vagy literál
címke	utasításcímke
IP	utasítás-számláló (16 bit)
SP	verem-mutató (16 bit)
BP	bázis-regiszter (16 bit)
(...)	a zárójelben megadott regiszter vagy memóriaterület tartalma
cím + 1:cím	a "cím"-en kezdődő, két egymást követő byte a memóriában
%	maradékképzés
&	logikai "és"
IOR	logikai "vagy" (megengedő)
XOR	logikai "kizáró vagy"
flag	16 bites regiszter, amelyben a jelzőbitek (flag-ek) találhatók
CF	átviteli (carry) flag
PF	paritás (parity) flag
AF	másodlagos átviteli (auxilary carry) flag
ZF	nulla (zero) flag
SF	előjel (sign) flag
TF	trap flag
IF	megszakítási (interrupt) flag
DF	irányt jelző (direction) flag
OF	túlcsordulási (overflow) flag

4.2 A 8086 és a 8088 utasításkészlete

AAA

Két négy bites nem tömörített (felső 4 bit nulla) BCD szám összeadása után, ha az eredmény AL-ben van, AL értékét átalakítja nem tömörített BCD számmá (számokká).

A processzor lépései:

```
ha  ((AL) & 0FH) > 9 vagy (AF) = 1 akkor
    (AL) = (AL) + 6
    (AH) = (AL) + 1
    (AF) = 1
    (CF) = (AF)
    (AL) = (AL)&0FH
```

Állított flag-ek: AF, CF

Példa: AAA ; igazítás öszeadás után

AAD

Az AX regiszterben lévő nem tömörített, binárisan kódolt decimális (BCD) szám konvertálása bináris számmá. Az eredmény az AL regiszterbe kerül. Osztás előtt az osztandó konvertálására szokták használni.

A processzor lépései:

```
    (AL) = (AH)*0AH + (AL)
    (AH) = 0
```

Állított flag-ek: PF, SF, ZF

Példa: AAD ; binárissá konvertálás osztás előtt

AAM

Nem tömörített,előjel nélküli BCD operandusokkal végzett szorzás AL-ben keletkező bináris eredményének konvertálása binárisan kódolt decimális számmá. Az eredmény az AX regiszterbe kerül.

A processzor lépései:

```
    (AH) = (AL)/0AH
    (AL) = (AL)%0AH     %: maradék
```

Állított flag-ek: PF, SF, ZF

Példa: AAM ;BCD-vé konvertálás szorzás után

AAS

Nem tömörített BCD operandusokkal végzett kivonás AL regiszterben keletkező eredményének nem tömörített átalakítása BCD számmá.

A processzor lépései:

```
ha   ((AL) & 0FH) > 9 vagy (AF) = 1 akkor
     (AL) = (AL)-6
     (AH) = (AH)-1
     (AF) = 1
     (CF) = (AF)
     (AL) = (AL)&0FH
```

Állított flag-ek: AF,CF

Példa: AAS ; igazítás kivonás után

ADC *akku,konst*

Konstans vagy literál hozzáadása az akkumulátor tartalmához az átviteli (carry) flag állításával.

A processzor lépései:

```
ha   (CF) = 1 akkor (akku) = (akku) + konst + 1
ha   (CF) = 0 akkor (akku) = (akku) + konst
```

Állított flag-ek: AF, CF, OF, SF, ZF

Példa: ADC AX,123

ADC *ált,konst*

Konstans vagy literál hozzáadása egy általános operandushoz az átviteli flag állításával.

A processzor lépései:

```
ha   (CF) = 1 akkor (ált) = (ált) + konst + 1
ha   (CF) = 0 akkor (ált) = (ált) + konst
```

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: ADC mem-cím,321

ADC *ált,reg*

Regiszter tartalmának hozzáadása egy általános operandushoz az átviteli flag állításával.

A processzor lépései:

```
ha   (CF) = 1 akkor (ált) = (ált) + (reg) + 1
ha   (CF) = 0 akkor (ált) = (ált) + (reg)
```

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: ADC mem-cím,CX

ADC *reg,ált*

Általános operandus hozzáadása egy regiszter tartalmához az átviteli flag állításával.

A processzor lépései:

ha (CF) = 1 akkor (reg) = (reg) + (ált) + 1
ha (CF) = 0 akkor (reg) = (reg) + (ált)

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: ADC CX,mem-cím

ADD *akku,konst*

Konstans vagy literál hozzáadása az akkumulátor tartalmához.

A processzor lépése:

(akku) = (akku) + konst

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: ADD AL,2

ADD *ált,konst*

Konstans vagy literál hozzáadása egy általános operandushoz.

A processzor lépése:

(ált) = (ált) + konst

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: ADD mem-cím,34

ADD *ált,reg*

Regiszter tartalmának hozzáadása egy általános operandushoz.

A processzor lépése:

(ált) = (ált) + (reg)

Állított flag-ek: AF, CF, OF, PF,SF, ZF

Példa: ADD mem[BP][DI],BX

ADD *reg,ált*

Általános operandus hozzáadása egy regiszter tartalmához.

A processzor lépése:

(reg) = (reg) + (ált)

Állított flag-ek: AF, CF, OF, PF,SF, ZF

Példa: ADD DX,szó[BX][DI]

AND *akku,konst*

Logikai "és" művelet az akkumulátor tartalma és egy konstans vagy literál között. Az eredmény az akkumulátorba kerül.

A processzor lépései:

(akku) = (akku)&konst
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: AND AX,2B4AH

AND *ált,konst*

Logikai "és" művelet egy általános operandus és egy konstans vagy literál között. Az eredmény felülírja az általános operandus eredeti értékét.

A processzor lépései:

(ált) = (ált)&konst
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: AND mem-byte,23H

AND *ált,reg*

Logikai "és"művelet egy általános operandus és egy regiszter tartalma között. Az eredmény felülírja az általános operandus eredeti értékét.

A processzor lépései:

(ált) = (ált)&(reg)
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: AND mem-szó,BP

AND *reg,ált*

Logikai "és" művelet egy regiszter tartalma és egy általános operandus között. Az eredmény a regiszterbe kerül.

A processzor lépései:

(reg) = (reg)&(ált)
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: AND BX,mem-szó

CALL *címke*

A "címke" által meghatározott szubrutin hívása. A címke a hívó szegmensen belül vagy egy másik szegmensben definiált. Szegmensen belüli híváskor:IP verembe kerül, a hívási cím = IP + előjeles eltolási cím. Szegmensek közötti híváskor először a CS majd az IP kerül a verembe; a hívásnak megfelelően beáll az új CS és IP érték.

A processzor lépései:

1. ha a címke a hívó szegmensen kívül van, akkor
   (SP) = (SP)-2
   ((SP) + 1:(SP)) = (CS)
   (CS) = SEG
2. (SP) = (SP)-2
3. ((SP) + 1:(SP)) = (IP)
4. (IP) = DEST

Hívó szegmensen belüli címke esetén:

DEST = (IP) + disp (disp: a címke távolsága a CALL utasítás végétől)

Hívó szegmensen kívüli címke esetén:

DEST = offset, SEG = szegmensszám

Állított flag-ek: -

Példák:
CALL near-label
CALL near-proc
CALL far-label
CALL far-proc

CALL *ált*

Indirekt szubrutinhívás. A hívott rutin a hívó szegmensen belül vagy egy másik szegmensben definiált.

A processzor lépései ugyanazok, mint direkt szubrutinhívás esetén, de a következő behelyettesítéseket kell figyelembe venni:

Ha a rutin a hívó szegmensen kívül van:

DEST = (ált), SEG = (ált + 2)

Ha a rutin a hívó szegmensen belül van:

DEST = (ált)

CBW

Byte konvertálása szóvá: az AL regiszter 7.bitjének értékével töltődik fel az AH regiszter minden bitje. Egy byte-os előjeles számok 2 byte-os előjeles számmá való átalakítása, egy byte-os előjeles osztási műveletek előtt.

Állított flag-ek: -

CLC

Az átviteli flag (carry) törlése.

A processzor lépése:

(CF) = 0

Állított flag: CF

CLD

Az irányt jelző flag törlése. Az SI és/vagy DI regisztereket változtató ciklus utasítások, ha a direction flag nulla (CLD utasítás után) az SI,DI regisztereket eggyel növelik. Ha a flag értéke 1, akkor eggyel csökkentik.

A processzor lépése:

(DF) = 0

Állított flag: DF

CLI

Hatására a maszkolható hardware megszakítások letiltásra kerülnek, az interrupt flag értéke nulla. Az utasítás a software interrupt-ra és az NMI-re nincs hatással.

A processzor lépése:

(IF) = 0

Állított flag: IF

CMC

Az átviteli (carry) flag átállítása az eredeti érték ellentettjére.

A processzor lépése:

ha (CF) = 1-(CF)

Állított flag: CF

CMP *akku,konst*

Az akkumulátor tartalmának összehasonlítása egy konstanssal vagy literállal. Az operandusok értéke nem változik. Az utasítás tulajdonképpen kivonja az akku értékét a konst-ból, az eredmény azonban nem jelenik meg, csupán az eredménynek megfelelő jelzőbitek állnak be.

A processzor lépése:

(akku)-konst

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: CMP AL,4

CMP *ált,konst*

Általános operandus összehasonlítása egy konstanssal vagy literállal. Az operandusok értéke nem változik.

A processzor lépése:

(ált)-konst

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: CMP mem-szó,112

CMP *ált,reg*

Általános operandus összehasonlítása egy regiszter tartalmával. Az operandusok értéke nem változik.

A processzor lépése:

(ált)-(reg)

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: CMP mem-szó,BX

CMP *reg,ált*

Regiszter tartalmának összehasonlítása egy általános operandussal. Az operandusok értéke nem változik.

A processzor lépése:

(reg)-(ált)

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: CMP CX,tár[BP][SI]

CMPS *string1, string2* vagy **CMPSB** vagy **CMPSW**

Karaktersorozatok összehasonlítása. Az operandusok értéke változatlan marad. String1 címe DS:SI-ben, String2 címe ES:DI-ben van. A flagek itt is beállnak.

A processzor lépései:

(LSRC)-(RSRC)
ha (DF) = 0 akkor
(SI) = (SI) + DELTA

```
        (DI) = (DI) + DELTA
egyébként
        (SI) = (SI)-DELTA
        (DI) = (DI)-DELTA
```

Byte-os műveletek esetében:

```
        LSRC = (SI), RSRC = (DI), DELTA = 1
```

Szavas műveletek esetében:

```
        LSRC = (SI) + 1:(SI), RSRC = (DI) + 1:(DI), DELTA = 2
```

```
Példa: MOV SI, OFFSET string1
       MOV DI, OFFSET string2
       CMPS string1,string2 vagy
       CMPS DS:BYTE PTR[SI],ES:[DI] vagy
       CMPSB
```

CWD

Szó konvertálása dupla szóvá: az AX regiszter 15. bitjének értékével töltődik fel a DX regiszter minden bitje. Előjeles, két byte-os osztások előtt használjuk.

A processzor lépése:

```
ha   (AX) < 8000h akkor (DX) = 0 egyébként (DX) = FFFFH
```

Állított flag: -

DAA

AL regiszter tartalmát tömörített BCD számokká alakítja, BCD számok összeadása után.

A processzor lépései:

```
ha   (AL)&0FH > 9 vagy (AF) = 1 akkor
     (AL) = (AL) + 6
     (AF) = 1
ha   (AL) > 9FH vagy (CF) = 1 akkor
     (AL) = (AL) + 60H
     (CF) = 1
```

Állított flag-ek: AF, CF, PF, SF, ZF

DAS

AL regiszter tartalmát tömörített BCD számokká alakítja BCD számok kivonása után.

A processzor lépései:

```
ha   (AL)&0FH > 9 vagy (AF) = 1 akkor
     (AL) = (AL)-6
     (AF) = 1
```

```
ha    (AL) > 9FH vagy (CF) = 1 akkor
      (AL) = (AL)-60H
      (CF) = 1
```

Állított flag-ek: AF, CF, PF, SF, ZF

DEC *ált*

Egy általános operandus értékének csökkentése 1-el.

A processzor lépése:

```
      (ált) = (ált)-1
```

Állított flag-ek: AF, OF, PF, SF, ZF

Példa: DEC mem-szó

DIV *ált*

Előjel nélküli bináris számok osztása. 2 byte-os osztáskor az osztandót a DX,AX regiszterpár tartalmazza, amelyet az utasításban megadott operandussal osztunk. A hányados AX-be, a maradék DX-be kerül. Byte-os osztáskor az osztandót AX tartalmazza, az osztó az utasítás operandusa. A hányados AL-be, a maradék AH-ba kerül.

A processzor lépései:

```
      (temp) = (NUMR)
ha    (temp)/(DIVR) > MAX akkor
      (QUO),(REM) definiálatlan
      (SP) = (SP)-2
      ((SP) + 1:(SP)) = flag-ek
      (IF) = 0
      (TF) = 0
      (SP) = (SP)-2
      ((SP) + 1:(SP)) = (CS)
      (CS) = (2) ; a 2-es es 3-as hely tartalma
      (SP) = (SP)-2
      ((SP) + 1:(SP)) = (IP)
      (IP) = (0) ; a 0-ás és 1-es hely tartalma (INT 0 generálódik)
egyébként
      (QUO) = (temp)/(DIVR)
      (REM) = (temp)%(DIVR)
```

Byte-os osztás esetén:
NUMR = AX, DIVR = ált, QOU = AL,
REM = AH, MAX = FFH

Szavas osztás esetén:
NUMR = DX:AX, DIVR = ált,QUO = AX,
REM = DX, MAX = FFFFH

Állított flag-ek: nincs érvényes flag

Példa: MOV AX,osztandó
DIV osztó

ESC *konst,ált*

Az utasítás a más processzorokkal (pl:matematikai koprocesszor) való kapcsolattartást biztosítja: a megcímzett tárterület tartalmát az adatbuszra teszi, de nem végez el vele műveletet. Az ESC mnemonik-ot követi egy opkód és egy utasítás is a koprocesszor számára.

HLT

Az utasítás hatására a programfutás felfüggesztődik. A processzor nem maszkolható megszakítás vagy reset hatására folytatja a program végrehajtását.

IDIV *ált*

Előjeles bináris számok osztása. Két byte-os osztáskor az osztandót a DX,AX regiszterpár tartalmazza, amelyet az utasításban megadott operandussal osztunk. A hányados AX-be, a maradék DX-be kerül. Negatív osztandó esetén használjuk előtte a CWD utasítást. Byte-os osztáskor az osztandót az AX tartalmazza, az osztó az utasítás operandusa. A hányados AL-be, a maradék AH-ba kerül. Negatív osztandó esetén használjuk előtte a CBW utasítást.

A processzor lépései:

```
     (temp) = (NUMR)
ha   (temp)/(DIVR) > 0 és (temp)/(DIVR) > MAX vagy
     (temp)/(DIVR) < 0 és (temp)/(DIVR) < 0-MAX-1 akkor
     (QUO),(REM) definiálatlan
     (SP) = (SP)-2
     ((SP) + 1:(SP)) = flag-ek
     (IF) = 0
     (TF) = 0
     (SP) = (SP)-2
     ((SP) + 1:(SP)) = (CS)
     (CS) = 2
     (SP) = (SP)-2
     ((SP) + 1:(SP)) = (IP)
     (IP) = (0) (INT 0 generálódik!)egyébként
     (QUO) = (temp)/(DIVR)
     (REM) = (temp)%(DIVR)
```

Byte-os osztás esetén:
NUMR = AX, DIVR = ált,QUO = AL,
REM = AH,MAX = FFH

Szavas osztás esetén:
NUMR = DX:AX,DIVR = ált,QUO = AX,
REM = DX,MAX = FFFFH.

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: MOV AX,osztandó[BX]
IDIV osztó

IMUL *ált*

Előjeles bináris számok szorzása. Szavas szorzáskor az egyik tényezőt az AX regiszter tartalmazza, a másik tényező az utasítás operandusa. A szorzat a DX,AX regiszterpárba kerül. Byte-os szorzáskor az egyik tényezőt AL tartalmazza, a másikat az utasítás operandusa. A szorzat AX-be kerül.

A processzor lépései:

```
       (DEST) = (LSRC)*(RSRC)
ha     (EXT) = (LOW) előjelkiterjesztése, akkor (CF) = 0
egyébként
       (CF) = 1
       (OF) = (CF)
```

Byte-os szorzás esetén:
LSRC = AL, RSRC = ált, DEST = AX,
EXT = AH, LOW = AL

Szavas szorzás esetén:
LSRC = AX, RSRC = ált, DEST = DX:AX,
EXT = DX, LOW = AX

Állított flag-ek: CF, OF

Példa: MOV AX,tényező1-szó
IMUL tényező2-szó

IN *akku,DX*

Byte vagy szó beolvasása AL-be vagy AX-be a DX regiszter által meghatározott I/O portról.

A processzor lépése:

(akku) = ((DX)port)

Állított flag-ek: -

Példa: IN AX,DX

IN *akku,konst*

Byte vagy szó beolvasása AL-be vagy AX-be a konstans vagy literál által megadottcímű i/o portról.

A processzor lépése:

(akku) = (konst port)

Állított flag-ek: -

Példa: IN AL,port-byte

INC *ált*

Az operandus értékének 1-el való növelése.

A processzor lépése:

(ált) = (ált) + 1

Állított flag-ek: AF, OF, PF, SF, ZF

Példa: INC mem-szó[BX]

INT 3

1 byte-os speciális software interrupt.

A processzor lépései:

(SP) = (SP)-2
((SP) + 1:(SP)) = flag-ek
(IF) = 0
(TF) = 0
(SP) = (SP)-2
((SP) + 1:(SP)) = (CS)
(CS) = (3*4 + 2)
(SP) = (SP)-2
((SP) + 1:(SP)) = (IP)
(IP) = (3*4)

Állított flag-ek: IF, TF

Példa: INT 3 ; egy byte: (11001100)

INT *konst*

Software interrupt 0-255 típuskóddal. A 0-255 közötti értékek az ugrási cím vektor táblájára mutatnak. A tábla effektív címe: 00000H-003FFH. Az operációs rendszer kezeli.

A processzor lépései:

(SP) = (SP)-2
((SP) + 1:(SP)) = flag-ek
(IF) = 0
(TF) = 0
(SP) = (SP)-2
((SP) + 1:(SP)) = (CS)

(CS) = (konst*4 + 2)
(SP) = (SP)-2
((SP) + 1:(SP)) = (IP)
(IP) = (konst*4)

Állított flag-ek: IF, TF

Példa: INT 44 ; két byte: (11001101 00101100)

INTO

Software interrupt, ha az overflow flag értéke 1.

A processzor lépései:

```
ha  (OF) = 1 akkor
    (SP) = (SP)-2
    ((SP) + 1:(SP)) = flag-ek
    (IF) = 0
    (TF) = 0
    (SP) = (SP)-2
    ((SP) + 1:(SP)) = (CS)
    (CS) = (12H)
    (SP) = (SP)-2
    ((SP) + 1:(SP)) = (IP)
```

Állított flag-ek: IF,TF ha túlcsordulás volt

INTR

Maszkolható külső interrupt. Csak a 8088 mikroprocesszor ismeri

IRET

Visszatérés interrupt-ból. IP és CS értéke és a flag-ek visszatöltődnek a veremből. (Az INT utasítások CS és IP aktuális értékét a verembe mentik, utána ugranak el a vektortábla által meghatározott címre.)

A processzor lépései:

```
    (IP) = ((SP) + 1:(SP))
    (SP) = (SP) + 2
    (CS) = ((SP) + 1:(SP))
    (SP) = (SP) + 2
    flag-ek = ((SP) + 1:(SP))
    (SP) = (SP) + 2
```

Állított flag-ek: minden flag

JA *címke* vagy **JNBE** *címke*

Ugrás a címkére ha a carry flag és a zero flag értéke nulla. A vezérlést csak 128 byte-nal kisebb távolságra lévő címkére lehet átadni. Az ugrás előjeles.

A processzor lépése:

ha (CF) IOR (ZF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JA közeli-címke
JNBE közeli-címke

JAE *címke* vagy **JNB** *címke*

Ugrás a címkére, ha a carry flag értéke nulla. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (CF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JNB közeli-címke
JAE közeli-címke

JB *címke* vagy **JNAE** *címke* vagy **JC** *címke*

Ugrás a címkére, ha a carry flag értéke 1. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (CF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JB közeli-címke
JNAE közeli-címke
JC közeli-címke

JBE *címke* vagy **JNA** *címke*

Ugrás a címkére ha a carry vagy a zero flag értéke 1. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (CF) IOR (ZF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JBE közeli-címke
JNA közeli címke

JCXZ *címke*

Ugrás a címkére, ha a CX regiszter tartalma nulla. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (CX) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JCXZ közeli címke

JE *címke* vagy **JZ** *címke*

Ugrás a címkére, ha a zero flag értéke 1. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (ZF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JE közeli címke
JZ közeli címke

JG *címke* vagy **JNLE** *címke*

Ugrás a címkére, ha az összehasonlítás eredménye "nagyobb". A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Az ugrás jelző feltétele: ((előjel flag XOR owerflow flag) AND zero flag) = 0. Előjeles ugrás.

A processzor lépése:

ha ((SF) XOR (OF)) vagy (ZF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JG közeli címke
JNLE közeli címke

JGE *címke* vagy **JNL** *címke*

Ugrás a címkére, ha az összehasonlítás eredménye "nagyobb vagy egyenlő". A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Az ugrás jelző feltétele: (előjel flag XOR overflow flag) = 0.Előjeles ugrás.

A processzor lépése:

ha (SF) XOR (OF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JGE közeli címke
JNL közeli címke

JL *címke* vagy **JNGE** *címke*

Ugrás a címkére, ha az összehasonlítás eredménye "kisebb". A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Az ugrás jelző feltétele: (előjel flag XOR overflow flag) = 1. Előjeles ugrás.

A processzor lépése:

ha (SF) XOR (OF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JL közeli címke
JNGE közeli címke

JLE *címke* vagy **JNG** *címke*

Ugrás a címkére, ha az összehasonlítás eredménye "kisebb vagy egyenlő". A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Az ugrás jelző feltétele:

((előjel flag XOR overflow flag)OR zero flag) = 1. Előjeles ugrás.

A processzor lépése:

ha (SF) XOR (OF) IOR (ZF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JLE közeli címke
JNG közeli címke

JMP *címke*

Feltétel nélküli direkt ugrás a címkére.

A processzor lépése:

ha a címke másik szegmensben van, akkor
(CS) = SEG
(IP) = DEST

Szegmensen belüli vezérlésátadás esetén:
(IP) = (IP) + disp

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága. Ha ez a távolság 128 byte-nál kisebb, akkor a cím kiszámítása előjel-kiterjesztéssel történik 16 bitre.

Példa: JMP közeli címke

Ha a vezérlést egy másik szegmensben levő címkére akarjuk átadni, akkor

DEST = offset, SEG = szegmenscím

Példa: JMP FAR PTR nyomtat

Állított flag-ek:-

JMP *ált*

Feltétel nélküli indirekt ugrás az operandus által kijelölt címre.

A processzor lépése:

ha a cím másik szegmensben van, akkor
(CS) = SEG
(IP) = DEST

Szegmensen belüli vezérlésátadás esetén
(IP) = (ált)

Példa: JMP szavas-változó

Ha a vezérlést egy másik szegmensben lévő címre akarjuk átadni, akkor

DEST = (ált)
SEG = (ált + 2)

Példa: JMP duplaszavas-változó

JNC *címke*

Ugrás a címkére, ha az átvitel (carry) flag értéke nulla. Előjeles ugrás.

A processzor lépése:

ha (CF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

Példa: JNC közeli címke

JNE *címke*

Ugrás a címkére, ha a zero flag értéke nulla. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (ZF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JNE közeli-címke
JNZ közeli-címke

JNO *címke*

Ugrás a címkére, ha az overflow flag értéke nulla. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (OF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JNO közeli-címke

JNP *címke* vagy JPO *címke*

Ugrás a címkére, ha a parity flag értéke nulla. A vezérlést csak 128 byte-nál kisebb távolságra lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (PF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek:

Példa: JNP közeli-címke
JPO közeli-címke

JNS *címke*

Ugrás a címkére, ha az előjel flag értéke nulla. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (SF) = 0 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JNS közeli-címke

JO *címke*

Ugrás a címkére, ha az overflow flag értéke 1. A vezérlést 128 byte-nál kisebb távoságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (OF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JO közeli-címke

JP *címke* vagy **JPE** *címke*

Ugrás a címkére, ha a parity flag értéke 1. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (PF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JP közeli-címke
JPE közeli-címke

JS *címke*

Ugrás a címkére,ha az előjel flag értéke 1. A vezérlést csak a 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Előjeles ugrás.

A processzor lépése:

ha (SF) = 1 akkor
(IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)

ahol "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: JS közeli-címke

LAHF

A flag-ek regisztere alacsonyabb helyiértékű 8 bitjének az AH regiszterbe való töltése. Az AH regiszter 7., 6., 4., 2. és 0.bitjébe kerülnek az előjel, zéró, auxiliary carry, paritás és carry flagek.

A processzor lépése:

(AH) = (SF):(ZF):X:(AF):X:(PF):X:(CF)

X azt jelenti, hogy AH megfelelő helyiértékű bitje változatlan marad.

Állított flag-ek: -

LDS *reg,ált*

A második operandussal meghatározott memóriaszó tartalom az első operandussal megadott regiszterbe, az azt követő memóriaszó tartalma a DS regiszterbe töltődik.

A processzor lépései:

(reg) = ((ált))
(DS) = ((ált) + 2)

Állított flag-ek: -

Példa: LDS BX,tábla[SI]

LEA *reg, ált*

A második operandus által megadott relatív címet az első operandussal meghatározott regiszterbe tölti.

A processzor lépése:

(reg) = Effektív cím

Állított flag-ek: -

Példa: LEA DX,változó

LES *reg,ált*

Az utasítás a megcímzett memóriabeli szót az adott regiszterbe, a következő szót az ES regiszterbe tölti.

A processzor lépései:

(reg) = ((ált))
(ES) = ((ált) + 2))

Állított flag-ek: -

Példa: LES BX,tábla[SI]

LOCK

A busz zárolása a következő utasítás végrehajtásának idejére. Célja, hogy az adott utasítás végrehajtása alatt a memóriához más processzor ne férhessen hozzá.

Állított flag-ek: -

Példa:
```
MOV AL,1                ;AL1-re állítása (azt jelenti, hogy zárolt)
LOCK XCHG SEMO,AL       ;tesztelés és zárolás
TEST AL,AL              ;a flag-ek beállítása AL szerint
JNZ ell                 ;ismétlés, ha zárolt
.
.
.
MOV SEMO,0              ;a zárolás feloldása
```

LODS *forrás* vagy **LODSB** vagy **LODSW**

A DS:SI regiszterpárral megcímzett byte vagy szó tartalmát betölti az AL vagy AX regiszterbe. SI tartalma a direction flag állásától függően automatikusan módosul. (Lásd még a CLD, STD utasításokat.)

A processzor lépései:

(DEST) = (SRC)

```
ha   (DF) = 0 akkor (SI) = (SI) + DELTA
ha   (DF) = 1 akkor (SI) = (SI)-DELTA
```

Szavas művelet esetében:

SRC = (SI) + 1:(SI),DEST = AX,DELTA = 2

Byte-os művelet esetében:

SRC = (SI),DEST = AL,DELTA = 1

Állított flag-ek: -

```
Példa:CLD        ;törli az irányt jelző flag-et, így SI inkrementálódni fog
      MOV        ;SI,OFFSET byte-string
      LODS       ;byte-string ;SI = SI + 1
      .
      .
      .
      STD        ;DF-et úgy állítja be, hogy SI dekrementálódni fog
      MOV        ;SI OFFSET szó-string
      LODS       ;szó-string ;SI = SI-2
```

LOOP *címke*

Ciklusszervező utasítás. Dekrementálja a CX regiszter tartalmát, és ha CX tartalma nem nulla, vezérlésátadást hajt végre a címkére. A vezérlést csak 128 byte-nál kisebb távolságra lévő címkére lehet átadni. Vigyázat, CX értékének vizsgálata előtt történik CX dekrementálása! Lásd még JCXZ utasítást.

A processzor lépései:

```
     (CX) = (CX)-1
ha   (CX) < > 0 akkor
     (IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)
```

ahol a "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: egy nem-nulla tömb 16 bites ellenőrző összegének kiszámítása.

```
      MOV CX,LENGTH tömb
      MOV AX,0
      MOV SI,AX
      ;
next: ADD AX,tömb[SI]
      ADD SI,TYPE tömb
      LOOP next
      MOV cks,AX
```

LOOPZ *címke* vagy **LOOPE** *címke*

Ciklusszervező utasítás. Dekrementálja a CX tartalmát; ha CX nem nulla és a zero flag értéke 1, ugrást hajt végre a 128 byte-nál kisebb távolságra lévő címkére.

A processzor lépései:

```
    (CX) = (CX)-1
ha  (ZF) = 1 & (CX) < >0 akkor
    (IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)
```

ahol a "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: byte-os tömb első, nullától különböző elemének megkeresése.

```
        MOV CX,LENGTH tömb
        MOV SI,-1
next:   INC SI
        CMP tömb [SI],0
        LOOPE next
        JNE talált
        . teendők, ha a tömb minden
        . eleme nulla

talált: ;SI tartalmazza az első, nullától különböző elem indexét
```

LOOPNZ *címke* vagy **LOOPNE** *címke*

Ciklusszervező utasítás. Dekrementálja a CX tartalmát; ha CX nem nulla és a zero flag értéke nulla, ugrást hajt végre a 128 byte-nál kisebb távolságra lévő címkére.

A processzor lépései:

```
    (CX) = (CX)-1

ha  (ZF) = 0 & (CX) < >0 akkor
    (IP) = (IP) + disp (előjelkiterjesztéssel 16 bitre)
```

ahol a "disp" az utasítás operandusaként megadott címkének az utasítás végétől való távolsága.

Állított flag-ek: -

Példa: 2 n byte hosszúságú vektor összegének kiszámítása. Az összegzést abbahagyjuk, ha mindkét összeadandó elem 0.

Ekkor a nullától különböző összegvektor hossza (SI)-1.

```
            MOV AX,0
            MOV SI,-1
            MOV CX,n
nem nulla:  INC SI
            MOV AL,vektor1[SI]
            ADD AL,vektor2[SI]
```

```
MOV ossz[SI],AL
LOOPNZ nem-nulla
```

MOV *akku,mem*

8-vagy 16-bites adat betöltése a memóriából az akkumulátorba.

A processzor lépése:

(akku) = (mem)

Állított flag-ek: -

Példa: MOV AL,mem-byte
MOV AX,mem-szó

MOV *mem,akku*

8-vagy 16-bites adat töltése az akkumulátorból a memóriába.

A processzor lépése:

(mem) = (akku)

Állított flag-ek: -

Példa: MOV mem-byte,AL
MOV mem-szó,AX

MOV *ált,konst*

8-vagy 16- bites konstans vagy literál betöltése regiszterbe vagy memóriába.

A processzor lépése:

(ált) = konst

Állított flag-ek: -

Példa: MOV mem-szó,1234

MOV *ált,reg*

Regiszter tartalmának betöltése regiszterbe vagy memóriába.

A processzor lépése:

(ált) = (reg)

Állított flag-ek: -

Példa: MOV mem-szó,BX

MOV *ált,szreg*

Szegmensregiszter tartalmának betöltése regiszterbe vagy memóriába.

A processzor lépése:

(ált) = (szreg)

Állított flag-ek: -

Példa: MOV vektor[BX][SI],SS

MOV *reg,konst*

Konstans vagy literál betöltése regiszterbe.

A processzor lépése:

(reg) = (konst)

Állított flag-ek: -

Példa: MOV CX,100

MOV *reg,ált*

Memória vagy regiszter tartalmának betöltése regiszterbe.

A processzor lépése:

(reg) = (ált)

Állított flag-ek: -

Példa: MOV AX,mem-szó

MOV *szreg,ált*

Memória vagy regiszter tartalmának töltése szegmensregiszterbe. (A szegmensregiszter nem lehet CS.)

A processzor lépése:

(szreg) = (ált)

Állított flag-ek: -

Példa: MOV ES,SS:új-szó[DI]

MOVS *fogadó,küldő* vagy MOVSB vagy MOVSW

Karaktersorozat másolása a küldő mezőből a fogadó mezőbe byte-onként vagy szavanként. A küldő cím DS:SI, a fogadó ES:DI.

A processzor lépései:

```
    (DEST) = (SCR)
ha  (DF) = 0 akkor
    (SI) = (SI) + DELTA
    (DI) = (DI) + DELTA egyébként
```

(SI) = (SI)-DELTA
(DI) = (DI)-DELTA

Byte-os művelet esetén:
SRC = (SI), DEST:(DI), DELTA = 1

Szavas művelet esetén:
SRC = (SI) + 1:(SI), DEST = (DI) + 1:(DI), DELTA = 2

Állított flag-ek: -

```
Példa: MOV SI,OFFSET küldő
       MOV DI,OFFSET fogadó
       MOV CX,LENGTH küldő
REP    MOVS fogadó,küldő
vagy
REP    MOVS ES:BYTE PTR[DI],DS:[SI]
vagy
REP    MOVSB
```

MOVSB

A DS:SI regiszterekkel megcímzett byte-ot átmásolja az ES:DI által meghatározott címre. Ha a REP prefixet is használjuk, akkor az ismétlési tényezőt CX tartalmazza.

MOVSW

A DS:SI által megcímzett szót másolja az ES:DI által meghatározott címre. Ha a REP prefixet is használjuk, akkor az ismétlési tényezőt CX tartalmazza.

MUL *ált*

Előjel nélküli bináris számok szorzása. Szavas szorzáskor az egyik tényezőt az AX regiszter tartalmazza, a másik tényező az utasítás operandusa. A szorzat a DX, AX regiszterpárba kerül. Byte-os szorzáskor az AL regiszter tartalmát szorozzuk az utasítás operandusával. A szorzat az AX regiszterbe kerül.

A processzor lépései:

(DEST) = (LSRC)*(RSRC)
ha (EXT) = 0 akkor
(CF) = 0 egyébként
(CF) = 1 , (OF) = (CF)

Byte-os műveletek esetén:
LSRC = AL, RSRC = ált, DEST = AX, EXT = AH

Szavas művelet esetén:
LSRC = AX, RSRC = ált, DEST = DX:AX, EXT = DX
Állított flag-ek: CF,OF

```
Példa: MOV AX,lsrc-szó
       MUL rsrc-szó ; az eredmény a DX:AX regiszterpárba kerül.
```

NEG *ált*

Az operandus kettes komplemensének előállítása. Az eredeti érték negáltja + 1.

A processzor lépései:

(ált) = SRC-(ált)
(ált) = (ált) + 1

Byte-os művelet esetén SRC = 0FFH

Szavas művelet esetén SRC = 0FFFFH

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: NEG mem-byte
NEG mem-szó

NMI

Megszakítás következik be, ha a külső megszakítás nem maszkolható interrupt (NMI). Csak a 8088 mikroprocesszor utasításkészlete tartalmazza ezt az utasítást.

NOP

Inkrementálja a PC regiszter tartalmát, egyébként nincs hatása. Kódja megegyezik a MOV AX,AX utasítással.

A processzor lépése:

(IP) = (IP) + 1

NOT *ált*

Az operandus minden bitjét az ellentettjére állítja.

A processzor lépése:

(ált) = SRC-(ált)

Byte-os művelet esetén SRC = 0FFH

Szavas művelet esetén SRC = 0FFFFH

Állított flag-ek: -

Példa: NOT mem-byte
NOT mem-szó

OR *akku,konst*

Logikai "vagy" művelet az akkumulátor tartalma és a konstans vagy literál között. Az eredmény az akkumulátorba kerül.

A processzor lépései:

(akku) = (akku) IOR konst
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: OR AX,123AH
OR AL,10110011B

OR *ált,konst*

Logikai "vagy" művelet memória vagy regiszter és konstans vagy literál között. Az eredmény az elsőként megadott operandusba kerül.

A processzor lépései:

(ált) = (ált) IOR konst
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: OR mem-byte,2FH
OR mem-szó,98B6H

OR *ált,reg*

Logikai "vagy" művelet memória vagy regiszter és regiszter között. Az eredmény az elsőként megadott operandusba kerül.

A processzor lépései:

(ált) = (ált) IOR (reg)
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: OR mem-byte,CL
OR mem-szó,DX

OR *reg,ált*

Logikai "vagy" művelet regiszter és regiszter vagy memória között. Az eredmény az elsőként megadott regiszterbe kerül.

A processzor lépései:

(reg) = (reg) IOR (ált)
(CF) = 0
(OF) = 0

Állított flag-ek:CF, OF, PF, SF, ZF

Példa: OR CH,mem-byte
OR AX,mem-szó

OUT DX,*akku*

Az akkumulátor tartalmát kiviszi a DX regiszter tartalma által megcímzett I/O portra.

A processzor lépése:

((DX)port) = (akku)

Állított flag-ek: -

Példa: OUT DX,AX

OUT *konst,akku*

Az akkumulátor tartalmát kiviszi az 1 byte-os konstans vagy literál által megcímzett I/O portra.

A processzor lépése:

(konst port) = (akku)

Állított flag-ek: -

Példa: OUT 44,AL

POP *ált*

A verem két felső byte-ját az utasításban megadott egyszavas operandusba tölti.

A processzor lépései:

(ált) = ((SP) + 1:(SP))
(SP) = (SP) + 2

Állított flag-ek: -

Példa: POP mem-szó

POP *reg*

A verem két felső byte-ját az utasításban megadott egyszavas regiszterbe tölti.

A processzor lépései:

(reg) = ((SP) + 1:(SP))
(SP) = (SP) + 2

Állított flag-ek: -

Példa: POP DX

POP *szreg*

A verem 2 felső byte-ját az utasításban megadott szegmensregiszterbe tölti. (A szegmensregiszter nem lehet CS.)

A processzor lépései:

(szreg) = ((SP) + 1:(SP))
(SP) = (SP) + 2

Állított flag-ek: -

Példa: POP SS

POPF

A verem 2 felső byte-ját a flag-ek regiszterébe tölti.

A processzor lépései:

flag-ek = ((SP) + 1:(SP))
(SP) = (SP) + 2

Állított flag-ek: minden flag

PUSH *ált*

Az utasításban megadott két byte-os operandus tartalmát a verem tetejére teszi.

A processzor lépései:

(SP) = (SP)-2
((SP) + 1:(SP)) = (ált)

Állított flag-ek: -

Példa: PUSH mem-szó

PUSH *reg*

Az utasításban megadott 16 bites regiszter tartalmát a verem tetejére teszi.

A processzor lépései:

(SP) = (SP)-2
((SP) + 1:(SP)) = (reg)

Állított flag-ek: -

Példa: PUSH CX

PUSH *szreg*

Az utasításban megadott szegmensregiszter tartalmát a verem tetejére teszi.

A processzor lépései:

```
        (SP) = (SP)-2
        ((SP) + 1:(SP)) = (szreg)
```

Állított flag-ek: -

Példa: PUSH ES

PUSHF

A flag-ek 16 bites tartalmát a verem tetejére teszi.

A processzor lépései:

```
        (SP) = (SP)-2
        ((SP) + 1:)SP)) = flag-ek
```

Állított flag-ek: -

RCL *ált*,1

Az operandus tartalmának ciklikus balra léptetése 1-gyel a carry flag-en keresztül.

A processzor lépései:

```
        (tmpfc) = (CF)
        (CF) = (ált) legmagasabb helyiértékű bitje
        (ált) = (ált)*2 + (tmpcf)
ha      (ált) legmagasabb helyiértékű bitje = (CF) akkor
        (OF) = 0
```

Állított flag-ek: CF,OF

Példa: RCL mem-byte,1

RCL *ált*,CL

Az operandus tartalmának ciklikus balra léptetése a carry flag-en keresztül annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
        (temp) = (CL)
amíg    (temp) < > 0
        (tmpcf) = (CF)
        (CF) = (ált) legmagasabb helyiértékű bitje
        (ált) = (ált)*2 + (tmpcf)
        (temp) = (temp)-1, majd
ha      (CL) = 1 akkor
        ha      (ált) legmagasabb helyiértékű bitje < > (CF) akkor
```

```
            (OF) = 1, egyébként (OF) = 0
egyébként (OF) definiálatlan.
```

Állított flag-ek: CF,OF

Példa: RCL változó[BX][DI],CL

RCR *ált*,1

Az operandus tartalmának ciklikus jobbra léptetése 1-gyel a carry flag-en keresztül.

A processzor lépései:

```
      (tmpcf) = (CF)
      (CF) = (ált) legalacsonyabb helyiértékű bitje
      (ált) = (ált)/2*
      (ált) legmagasabb helyiértékű bitje = (tmpfc)
ha    (ált) legmagasabb helyiértékű bitje < >
      (ált) köv. legmagasabb helyiértékű bitjével
      akkor (OF) = 1 egyébként (OF) = 0
```

Állított flag-ek: CF, OF

Példa: RCR mem-byte,1

RCR *ált*,CL

Az operandus tartalmának ciklikus jobbra léptetése a carry flag-en keresztül annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
      (temp) = (CL)
amíg  (temp) = 0
      (tmpcf) = (CF)
      (CF) = (ált) legalacsonyabb helyiértékű bitje
      (ált) = (ált)/2*
      (ált) legmagasabb helyiértékű bitje = (tmpcf)
      (temp) = (temp)-1, majd
ha    (CL) = 1 akkor
      ha    (ált) legmagasabb helyiértékű bitje =
            (ált) köv. legmagasabb helyiértékű bitjével
            akkor (OF) = 1 egyébként (OF) = 0
egyébként (OF) definiálatlan
```

REP

A prefixet követő karaktersorozatkezelő utasítás ismétlése, amíg a CX regiszter tartalma nem nulla.

A processzor lépései:

```
amíg (CX) < > 0
      függőben lévő megszakítás feldolgozása (ha létezik)
      karakterkezelő utasítás végrehajtása a köv. byte-on
      (CX) = (CX)-1
```

Állított flag-ek: ld. karakterkezelő utasítás

Példa: REP MOVS fogadó,küldő

REPE vagy REPZ

A prefixet követő karaktersorozatkezelő utasítás ismétlése, ameddig CX regiszter tartalma nem nulla, vagy a zero flag értéke nem egy.

A processzor lépései:

```
amíg (CX) < > 0
      függőben lévő megszakítás feldolgozása (ha létezik)
      karakterkezelő utasítás végrehajtása a köv. byte-on
      (CX) = (CX)-1
      ha az utasítás CMPS vagy SCAS és (ZF) = 1
      akkor kilépés a ciklusból.
```

Állított flag-ek: ld. karakterkezelő utasítás.

REPN vagy REPNZ

A prefixet követő karaktersorozatkezelő utasítás ismétlése, ameddig CX regiszter tartalma nem nulla, vagy zero flag értéke nem nulla.

A processzor lépései:

```
amíg (CX) < > 0
      függőben lévő megszakítás feldolgozása (ha létezik)
      karakterkezelő utasítás végrehajtása a
      köv. byte-on vagy szón
      (CX) = (CX)-1
      ha az utasítás CMPS vagy SCAS és (ZF) = 0
      akkor kilépés a ciklusból.
```

Állított flag-ek: ld. karakterkezelő utasítás.

RET

Szubrutinból való visszatérés.

A processzor lépései:

```
      (IP) = ((SP) + 1:(SP))
      (SP) = (SP) + 2
      ha külső szegmensből való visszatérés, akkor
```

```
        (CS) = ((SP) + 1:(SP))
        (SP) = (SP) + 2
```

Állított flag-ek: -

Példa: RET

RET *konst*

Szubrutinból való visszatérés. A verem értékhez a konstans értéke is hozzáadódik, így alakul ki a végleges cím.

A processzor lépései:

```
        (IP) = ((SP) + 1:(SP))
        (SP) = (SP) + 2
        ha külső szegmensből való visszatérés, akkor
        (CS) = ((SP) + 1:(SP))
        (SP) = (SP) + 2
        ha az utasításnak van operandusa, akkor
        (SP) = (SP) + konst
```

Állított flag-ek: -

Példa: RET 12

ROL *ált*, 1

Az operandus tartalmának ciklikus balra léptetése 1-gyel.

A processzor lépései:

```
        (CF) = (ált) legmagasabb helyiértékű bitje
        (ált) = (ált)*2 + (CF)
ha      (ált) legmagasabb helyiértékű bitje < > (CF) akkor
        (OF) = 1
egyébként (OF) = 0
```

Állított flag-ek: CF, OF

Példa: ROL mem-byte,1

ROL *ált*,CL

Az operandus tartalmának ciklikus balra léptetése annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
        (temp) = (CL)
amíg    (temp) < > 0
        (CF) = (ált) legmagasabb helyiértékű bitje
        (ált) = (ált)*2 + (CF)
        (temp) = (temp)-1, majd
```

```
ha   (CL) = 1 akkor
     ha    (ált) legmagasabb helyiértékű bitje < > (CF) akkor
           (OF) = egyébként (OF) = 0
egyébként (OF) = meghatározatlan
```

Állított flag-ek: CF, OF

Példa: ROL szó,CL

ROR *ált*,1

Az operandus tartalmának ciklikus jobbra léptetése 1-gyel.

A processzor lépései:

```
     (CF) = (ált) legalacsonyabb helyiértékű bitje
     (ált) = (ált)/2
     (ált) legmagasabb helyiértékű bitje = (CF)
ha   (ált) legmagasabb helyiértékű bitje < >
     (ált) köv. legmagasabb helyiértékű bitjével
     akkor (OF) = 1 egyébként (OF) = 0
```

Állított flag-ek: CF, OF

Példa: ROR mem-szó,1

ROR *ált*,CL

Az operandus tartalmának ciklikus jobbra léptetése annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
     (temp) = (CL)
amíg (temp) < > 0
     (CF) = (ált) legalacsonyabb helyiértékű bitje
     (ált) = (ált)/2
     (ált) legmagasabb helyiértékű bitje = (CF)
     (temp) = (temp)-1, majd
ha   (CL) = 1 akkor
     ha    (ált) legmagasabb helyiértékű bitje < >
           (ált) köv. legmagasabb helyiértékű bitjével
           akkor (OF) = 1 egyébként (OF) = 0
egyébként (OF) definiálatlan
```

Állított flag-ek: CF,OF

Példa: ROR mem-szó,CL

SAHF

Az AH regiszter tartalmának betöltése a flag-ek regiszterének alacsonyabb helyiértékű 8 bitjére. A LAHF utasítás fordítottja.

A processzor lépése:

```
        (SF):(ZF):X:(AF):x:(PF):X:(CF) = (AH)
```

X azt jelenti, hogy a flag-ek regiszterének megfelelő helyiértékű bitje változatlan marad.

Állított flag-ek: AF, CF, PF, SF, ZF

SAL *ált*,1 vagy SHL *ált*,1

Az operandus tartalmának artimetikai balra léptetése 1-gyel.

A processzor lépései:

```
        (CF) = (ált) legmagasabb helyiértékű bitje
        (ált) = (ált)*2
ha      (ált) legmagasabb helyiértékű bitje < > (CF) akkor
        (OF) = 1 egyébként (OF) = 0
```

Állított flag-ek: CF, OF PF, SF, ZF

Példa: SAL változó,1
       SHL mem-byte,1

SAL *ált*,CL vagy SHL *ált*,CL

Az operandus tartalmának artimetikai balra léptetése annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
        (temp) = (CL)
amíg    (temp) < > 0
        (CF) = (ált) legmagasabb helyiértékű bitje
        (ált) = (ált)*2
        (temp) = (temp)-1, majd
ha      (CL) = 1 akkor
        ha      (ált) legmagasabb helyiértékű bitje < > (CF) akkor
                (OF) = 1, egyébként
egyébként (OF) definiálaltlan
```

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: SAL AX,CL
       SHL mem-szó,CL

SAR *ált*,1

Az operandus tartalmának artimetikai jobbra léptetése 1-gyel.

A processzor lépései:

```
        (CF) = (ált) legalacsonyabb helyiértékű bitje
        (ált) = (ált)/2 előjeles osztás, lefelé kerekítéssel
```

```
ha    (ált) legmagasabb helyiértékű bitje < >
      (ált) köv. legmagasabb helyiértékű bitjével
      akkor (OF) = 1 egyébként (OF) = 0
```

Állított flag-ek: CF,OF

Példa: SAR mem-byte,1

SAR *ált*,CL

Az operandus tartalmának artimetikai jobbra léptetése annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
      (temp) = 1
amíg  (temp) < > 0
      (CF) = (ált) legalacsonyabb helyiértékű bitje
      (ált) = (ált)/2 (előjeles osztás, lefelé kerekítés)
      (temp) = (temp)-1
ha    (CL) = 1 akkor
      ha    (ált) legmagasabb helyiértékű bitje < >
            (ált) köv. legmagasabb helyiértékű bitjével,
      egyébként (OF) = 0
```

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: SAR AX, CL

SBB *akku,konst*

Konstans vagy literál és a carry flag kivonása az akkumulátor tartalmából. Az eredmény az akkumulátorba kerül.

A processzor lépései:

```
ha    (CF) = 1 akkor (akku) = (akku)-konst-1
ha    (CF) = 0 akkor (akku) = (akku)-konst.
```

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: SBB AX,432

SBB *ált,konst*

Konstans vagy literál és a carry flag kivonása egy memóriacím vagy egy regiszter tartalmából. Az eredmény az adott című területre vagy a regiszterbe kerül.

A processzor lépései:

```
ha    (CF) = 1 akkor (ált) = (ált)-konst-1
ha    (CF) = 0 akkor (ált) = (ált)-konst
```

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: SBB mem-szó,123

SBB *ált,reg*

Regisztertartalom és a carry flag kivonása regiszter vagy adott címû memóriaterület tartalmából. Az eredmény az elsőként felírt operandusba kerül.

A processzor lépései:

ha (CF) = 1 akkor (ált) = (ált)-(reg)-1
ha (CF) = 0 akkor (ált) = (ált)-(reg)

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: SBB mem-szó,AX

SBB *reg,ált*

Regisztertartalom vagy adott címû terület és a carry flag tartalmának kivonása egy regiszter tartalmából. Az eredmény az elsőként felírt operandusba kerül.

A processzor lépései:

ha (CF) = 1 akkor (reg) = (reg)-(ált)-1
ha (CF) = 0 akkor (reg) = (reg)-(ált)

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: SBB AX,mem-szó

SCAS *string*

Az ES:DI regiszterek tartalmával címzett byte vagy szó összehasonlítása az AL vagy AX regiszter tartalmával. A mûvelet mindkét operandusa változatlan marad, csak a flag-eket állítja.

A processzor lépései:

(LSRC)-(RSRC)
ha (DF) = 0 akkor (DI) = (DI) + DELTA
egyébként (DI) = (DI)-DELTA

Byte-os mûvelet esetén:
LSRC = AL, RSRC = (DI),DELTA = 1

Szavas mûvelet esetén
LSRC = AX,RSRC = (DI) + 1:(DI),DELTA = 2

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa:
```
CLD  ;törli DF-et. így DI inkrementálódik majd
MOV  DI OFFSET byte-string
MOV  AL, 'M'
```

```
     SCAS byte-string
vagy
     SCAS ES:BYTE PTR[DI]
vagy
     SCASB
     STD ;beállítja DF-et, így DI dekrementálódik majd
     MOV  DI OFFSET szó-string
     MOV  AX, 'MD'
     SCAS szó-string
```

SHL *ált*,1 vagy SAL *ált*,1

Az operandus tartalmának logikai balra léptetése 1-gyel.

A processzor lépései:

```
     (CF) = (ált) legmagasabb helyiértékű bitje
     (ált) = (ált)*2
ha   (ált) legmagasabb helyiértékű bitje < > (CF)
     akkor (OF) = 1 egyébként (OF) = 0
```

Állított flag-ek: CF, OF, PF, SF, ZF

```
Példa: SAL változó,1
       SHL mem-byte,1
```

SHL *ált*,CL vagy SAL *ált*,CL

Az operandus tartalmának logikai balra léptetése annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
     (temp) = (CL)
amíg (temp) < > 0
     (CF) = (ált) legmagasabb helyiértékű bitje
     (ált) = (ált)*2
     (temp) = (temp)-1, majd
 ha  (CL) = 1 akkor
     ha   (ált) legmagasabb helyiértékű bitje < > (CF)
          akkor (OF) = 1 egyébként (OF) = 0
egyébként (OF) definiálatlan.
```

Állított flag-ek: CF, OF, PF, SF, ZF

```
Példa: SAL AX,CL
       SHL mem-szó,CL
```

SHR *ált*,1

Az operandus tartalmának logikai jobbra léptetése 1-gyel.

A processzor lépései:

```
      (CF) = (ált) legalacsonyabb helyiértékű bitje
      (ált) = (ált)/2 (előjeles osztás, lefelé kerekítés)
ha    (ált) legmagasabb helyiértékű bitje < > (ált) köv legmagasabb helyiértékű bitjével
      akkor (OF) = 1
egyébként (OF) = 0
```

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: SHR mem-byte,1

SHR *ált*,CL

Az operandus tartalmának logikai jobbra léptetése annyival, amennyi a CL regiszter tartalma.

A processzor lépései:

```
      (temp) < > (CL)
amíg  (temp) = 0
      (CF) = (ált) legalacsonyabb helyiértékű bitje
      (ált) = (ált)/2 (előjeles osztás, lefelé kerekítés)
      (temp) = (temp)-1, majd
ha    (CL) = 1 akkor
      ha    (ált) legmagasabb helyiértékű bitje =
            (ált) köv. legmagasabb helyiértékű bitjével
            akkor (OF) = 1 egyébként OF = 0
egyébként (OF) definiálatlan
```

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: SHR mem-szó,CL

STC

Az átviteli (carry) flag 1-re állítása.

A processzor lépése:

```
      (CF) = 1
```

Állított flag: CF

STD

A feldolgozás irányát jelző (direktion) flag 1-re állítása. Ekkor az SI és/vagy DI regisztereket automatikusan állító utasítások autodecrement módban dolgoznak (pl. LODS, STOS)

A processzor lépése:

```
      (DF) = 1
```

Állított flag: DF

STI

A megszakítást engedélyező (interrupt) flag 1-re állítása.

A processzor lépése:

(IF) = 1

Állított flag: IF

STOS *string*, **STOSB, STOSW**

AL vagy az AX regiszter tartalmának tárolása az ES:DI regiszterek tartalmával megcímzett byte-on vagy szón. Az utasítás végrehajtása után DI értéke a direction flag állásától függően eggyel csökken vagy nő.

A processzor lépései:

(DEST) = (SCR)
ha (DF) = 0 akkor (DI) = (DI) + DELTA
egyébként (DI) = (DI)-DELTA

Byte-os művelet esetén:
SRC = AL,DEST = (DI),DELTA = 1

Szavas művelet esetén:
SRC = AX,DEST = (DI) + 1:(DI),DELTA = 2

Állított flag-ek: -

Példa: MOV DI,OFFSET byte-string
STOS BYTE-string vaagy STOSB
MOV DI,OFFSET szó-string
STOS szó-string vagy STOSW

SUB *akku,konst*

Konstans vagy literál kivonása az akkumulátor tartalmából. Az eredmény az akkumulátorba kerül.

A processzor lépése:

(akku) = (akku)-konst

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: SUB AL,3

SUB *ált,konst*

Konstans vagy literál kivonása egy adott című memóriaterület vagy regiszter tartalmából. Az eredmény az elsőként felírt operandusba kerül.

A processzor lépése:

(ált) = (ált)-konst

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: SUB mem-szó[DI][BX],1234

SUB *ált,reg*

Regisztertartalom kivonása egy adott című memóriaterület vagy regiszter tartalmából. Az eredmény az elsőként felírt operandusba kerül.

A processzor lépése:

(ált) = (ált)-(reg)

Állított flag-ek: AF, CF, OF, PF, SF, ZF

Példa: SUB mem-szó,CX

SUB *reg,ált*

Adott című memóriaterület vagy egy regiszter tartalmának kivonása egy regiszter tartalmából. Az eredmény az elsőként felírt operandusba kerül.

A processzor lépése:

(reg) = (reg)-(ált)

Állított flag-ek: AF, CF,OF, PF, SF, ZF

Példa: SUB BX,mem-szó

TEST *akku,konst*

Logikai "és" művelet konstans vagy literál és az akkumulátor tartalma között. A művelet operandusai nem változnak, a flag-eket állítja.

A processzor lépései:

(akku)&konst
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: TEST AL,2

TEST *ált,konst*

Logikai "és" művelet konstans vagy literál és egy adott című memóriaterület vagy egy regiszter tartalma között. A művelet operandusai nem változnak, a flag-eket állítja.

A processzor lépései:

(ált)&konst

(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: TEST mem-szó 532

TEST *ált,reg*

Logikai "és" művelet egy regiszter tartalma és egy adott című memóriaterület vagy egy regiszter tartalma között. A művelet operandusai nem változnak, a flag-eket állítja.

A processzor lépései:

(ált)&(reg)
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa TEST mem-szó,SI

TEST *reg,ált*

Logikai "és" művelet egy adott című memóriaterület vagy egy regiszter tartalma és egy regisztertartalom között. A művelet operandusai nem változnak, a flag-eket állítja.

A processzor lépései:

(reg)&(ált)
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: TEST DI,mem-szó

WAIT

Az utasítások végrehajtása addig szünetel, amíg a processzor "TEST" bemenet szintje nulla. Ilyenkor az egyik processzor várhat addig, amíg a másik bizonyos műveletet be nem fejez.

Állított flag-ek: -

XCHG *akku,reg*

Az akkumulátor és egy regiszter tartalmának megcserélése.

A processzor lépései:

o(temp) = (akku)
(akku) = (reg)
(reg) = (temp)

Állított flag-ek: -

Példa: XCHG AX,BX

XCHG *ált,reg*

Egy adott című memóriaterület vagy egy regiszter tartalmának megcserélése egy regiszter tartalmával.

A processzor lépései:

(temp) = (ált)
(ált) = (reg)
(reg) = (temp)

Állított flag-ek: -

Példa: XCHG mem-szó,CX

XCHG *reg,akku*

Egy regiszter tartalmának megcserélése az akkumulátor tartalmával.

A processzor lépései:

(temp) = (reg)
(reg) = (akku)
(akku) = (temp)

Állított flag-ek: -

Pelda: XCHG DX,AX

XCHG *reg,ált*

Egy regiszter tartalmának megcserélése egy adott című memóriaterület vagy egy regiszter tartalmával.

A processzor lépései:

(temp) = (reg)
(reg) = (ált)
(ált) = (temp)

Állított flag-ek: -

Példa: XCHG DX,mem-szó

XLAT *mem*

A DS:(BX) + (AL) regiszterek tartalma által megcímzett byte az AL regiszterbe kerül. Rendszerint akkor használjuk, ha táblázatból akarunk adatokat behozni AL-be. Ekkor DS:BX mutat a táblázat elejére, AL pedig indexként rámutat a táblázat adott elemére.

A processzor lépése:

(AL) = ((BX)) + (AL))

Állított flag-ek: -

Példa: MOV AL,5
MOV BX,OFFSET táblanév
XLAT táblanév

XOR *akku,konst*

"Kizáró vagy" művelet egy konstans vagy literál és az akkumulátor tartalma között. Az eredmény az akkumulátorba kerül.

A processzor lépései:

(akku) = (akku) XOR konst
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: XOR AX,512

XOR *ált,konst*

"Kizáró vagy" művelet egy konstans vagy literál és egy adott című memóriaterület vagy egy regiszter tartalma között. Az eredmény az elsőként felírt operandusba kerül.

A processzor lépései:

(ált) = (ált) XOR konst
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: XOR mem-byte,1DH

XOR *ált,reg*

"Kizáró vagy" művelet egy regiszter tartalma és egy adott című memóriaterület vagy egy regiszter tartalma között. Az eredmény az elsőként felírt operandusba kerül.

A processzor lépései:

(ált) = (ált) XOR (reg)
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: XOR mem-szó,BX

XOR *reg,ált*

"Kizáró vagy" művelet regiszter vagy memóriaterület tartalma és egy regiszter tartalma között. Az eredmény az elsőként felírt operandusba kerül

A processzor lépései:

(reg) = (reg) XOR (ált)
(CF) = 0
(OF) = 0

Állított flag-ek: CF, OF, PF, SF, ZF

Példa: XOR DX, mem-szó

4.3 A 8087 utasításkészlete

A 8087 lebegőpontos koprocesszor a 8086/8088 mikroprocesszor utasításkészletét bővíti. A 8087 lehetővé teszi a lebegőpontos és nagy pontosságot igénylő számítások elvégzését.

4.3.1 A 8087-es numerikus adat-processzor felépítése

Adat formátumok

adattípus	**bitek**
egész(WI) kettes kompl.	0 - 15
egész(SI) kettes kompl.	0 - 31
egész(LI) kettes kompl.	0 - 63
binárisan kódolt dec.(PD)	0 - 79
valós(SR)	0 - 31
valós(LR)	0 - 63
valós(TR)	0 - 79

Az egyes adattípusok felépítése bitenként:

WI: 0 - 14. érték, 15. előjel.
SI: 0 - 30. érték, 31. előjel.
LI: 0 - 62. érték, 63. előjel.
PD: 0 - 71. érték, 72-78. nem használt, 79. előjel.
SR: 0 - 22. alap, 23-30. kitevő, 31. előjel.
LR: 0 - 51. alap, 52-62. kitevő, 63. előjel.
TR: 0 - 63. alap, 64-78. kitevő, 79. előjel.

Az SR és LR adattípus esetében az alap legnagyobb helyiértékű bitje nincs ábrázolva, mert értéke mindig 1.

A regiszterek felépítése

A 8087-es matematikai koprocesszor 8 db., egyenként 80 bites regisztert használ az adatok kezelésére, plusz minden regiszter mellett egy 2 bites TAG regisztert, mely a belső regiszterek foglaltságát jelzi.

A regiszterek bit felosztása:

0 - 63. alap
64 - 78. kitevő
79. előjel bit

Egyéb regiszterek:

2 db 16 bites vezérlő regiszter
2 db 16 bites program számláló
2 db 16 bites adat mutató.

A vezérlő regiszterek felépítése:

Az egyik vezérlő regiszter input, a másik output üzemmódú.

Az **input** vezérlő regiszter felépítése:
0. bit: érvénytelen művelet
1. bit: denormalizált operandus
2. bit: nullával való osztás
3. bit: túlcsordulás
4. bit: pontosság
5. bit: nem használható
6. bit: megszakítás engedélyezés
7. - 8. bit: pontosság vezérlés
9. - 10. bit: kerekítés vezérlés
11. bit: ábrázolhatatlanul nagy szám kezelése
12. - 15. bit: nem használható

Az **output** (státusz) regiszter felépítése:
0. bit: érvénytelen művelet
1. bit: denormalizált operandus
2. bit: nullával való osztás
3. bit: túlcsordulás
4. bit: alulcsordulás
5. bit: pontosság
6. bit: nem használható
7. bit: megszakítás kérés
8. - 10. és a 14. bit: feltétel kódok
11 - 13. bit: verem (adat regiszterek) teteje mutató
15. bit: processzor foglaltságának jelzése

Utasítás készlet

Minden utasítás az ESC kóddal kezdődik, mivel innen "tudja" a koprocesszor, hogy a következő parancsok neki szólnak.

Az ESC kódja: 11011xxx, ahol xxx már az utasítás része.

A. Adatátviteli utasítások

FLD: értékek koprocesszor adatterületre töltése.
Általános forma:
(ESC mf 1)(mod 0 0 0 r/m)(LO byte)(HI byte)
Speciális forma:
Adat az LI regiszterbe:
(ESC 1 1 1)(mod 1 0 1 r/m)(LO byte)(HI byte)
Adat a TR regiszterbe:
(ESC 0 1 1)(mod 1 0 1 r/m)(LO byte)(HI byte)
Adat a PD regiszterbe:
(ESC 1 1 1)(mod 1 0 0 r/m)(LO byte)(HI byte)
Regiszter értéke a legfelső regiszterbe:
(ESC 0 0 1)(1 1 0 0 0 st(i))

FST: regiszterek memóriába töltése a megadott adattípus szerint (a legfelső regiszter értéke változatlan marad).
Általános forma:
(ESC mf 1)(mod 0 1 0 r/m)(LO byte)(HI byte)
Speciális forma:
A regiszter teteje másik regiszterbe:
(ESC 0 1 1)(1 1 0 1 0 st(i))

FSTP: ugyanaz, mint az FST, de a legfelső regiszter tartalma elvész.
Általános forma:
(ESC mf 1)(mod 0 1 1 r/m)(LO byte)(HI byte)
Speciális forma:
LI regiszter tartalmának memóriába töltése:
(ESC 1 1 1)(mod 1 1 1 r/m)(LO byte)(HI byte)
TR regiszter tartalmának memóriába töltése:
(ESC 0 1 1)(mod 1 1 1 r/m)(LO byte)(HI byte)
PD regiszter tartalmának memóriába töltése:
(ESC 1 1 1)(mod 1 1 0 r/m)(LO byte)(HI byte)
A regiszter teteje másik regiszterbe:
(ESC 0 0 1)(1 1 0 0 1 st(i))

FXCH: st(0) és st(i) tartalmának kicserélése:
(ESC 0 0)(1 1 0 0 1 ST(i))

B. Összehasonlító utasítások

FCOM: összehasonlítás (legfelső regiszter értéke változatlan marad).
Általános forma:
(ESC mf 0)(mod 0 1 0 r/m)(LO byte)(HI byte)
st(0) összehasonlítása st(i)-vel:
(ESC 0 0 0)(1 1 0 1 0 st(i))

FCOMP: összehasonlítás (a legfelső regiszter tartalma elvész).
Általános forma:
(ESC mf 0)(mod 0 1 1 r/m)(LO byte)(HI byte)
st(0) összehasonlítása st(i)-vel:
(ESC 0 0 0)(1 1 0 1 1 st(i))

FCOMPP: a legfelső két regiszter értékének összehasonlítása és elvesztése.
(ESC 1 1 0)(1 1 0 1 1 0 0 1)

FTST: a felső regiszter értékének összehasonlítása nullával.
(ESC 0 0 1)(1 1 1 0 0 1 0 0)

FXAM: az input vezérlő regiszter 0-3. bitjét állítja a felső regiszterben lévő adat típusa szerint.
(ESC 0 0 1)(1 1 1 0 0 1 0 1)

C. Aritmetikai utasítások

FADD: összeadás
Általános forma:
(ESC mf 0)(mod 0 0 0 r/m)(LO byte)(HI byte)
Speciális forma:
A legfelső regiszter összeadása egy másikkal:
(ESC d p 0)(1 1 0 0 0 ST(i))

FSUB: kivonás
Általános forma:
(ESC mf 0)(mod 1 0 R r/m)(LO byte)(HI byte)
Speciális forma:
regiszter tartalmának kivonása a legfelső regiszterből, vagy fordítva:
(ESC d p 0)(1 1 1 0 R ST(i))

FMUL: szorzás
Általános forma:
(ESC mf 0)(mod 0 0 1 r/m)(LO byte)(HI byte)
Speciális forma:
a legfelső regiszter szorzása egy másikkal:
(ESC d p 0)(1 1 0 0 1 ST(i))

FDIV: osztás
Általános forma:
(ESC mf 0)(mod 1 1 R r/m)(LO byte)(HI byte)
Speciális forma:
a legfelső regiszter osztása egy másikkal:
(ESC d p 0)(1 1 1 1 R ST(i))

D. Függvények

SQRT: a legfelső regiszter tartalmának négyzetgyöke.
(ESC 0 0 1)(1 1 1 1 1 0 1)

FSCALE: ST(0)*2|ST(1)
(ESC 0 0 1)(1 1 1 1 1 1 1 0)

FPREM: ST(0)/ST(1) maradéka
(ESC 0 0 1)(1 1 1 1 1 0 0 0)

FRNDINT: ST(0) kerekített egész értéke.
(ESC 0 0 1)(1 1 1 1 1 1 0 0)

FXTRACT: ST(0) alapja ST(0)-ban marad, a kitevő ST(1)-be kerül.
(ESC 0 0 1)(1 1 1 1 0 1 0 0)

FABS: ST(0) abszolút értékét adja.
(ESC 0 0 1)(1 1 1 0 0 0 0 1)

FCHS: előjel váltása.
(ESC 0 0 1)(1 1 1 0 0 0 0 0)

E. Trigonometrikus és logaritmus függvények

FPTAN: ST(0) parciális tangense.
(ESC 0 0 1)(1 1 1 1 0 0 1 0)

FPATAN: ST(0) parciális arcus tangense.
(ESC 0 0 1)(1 1 1 1 0 0 1 1)

F2XM1: 2|ST(0) mínusz 1
(ESC 0 0 1)(1 1 1 1 0 0 0 0)

FYL2X: ST(1)*log2(ST(0))
(ESC 0 0 1)(1 1 1 1 0 0 0 1)

FYL2XP1: ST(1)*log2(ST(0) + 1)
(ESC 0 0 1)(1 1 1 1 1 0 0 1)

F. Konstansok

FLDZ: ST(0) = 0
(ESC 0 0 1)(1 1 1 0 1 1 1 0)

FLD1: ST(0) = 10
(ESC 0 0 1)(1 1 1 0 1 0 0 0)

FLDPI: ST(0) = PI értékével.
(ESC 0 0 1)(1 1 1 0 1 0 1 0)

FLDL2T: ST(0) = log2(10)
(ESC 0 0 1)(1 1 1 0 1 0 0 1)

FLDL2E: ST(0) = log2(e)
(ESC 0 0 1)(1 1 1 0 1 0 1 0)

FLDLG2: ST(0) = log10(2)
(ESC 0 0 1)(1 1 1 0 1 1 0 0)

FLDLN2: ST(0) = loge(2)
(ESC 0 0 1)(1 1 1 0 1 1 0 1)

Vezérlő utasítások

FINIT: 8087 inicializálása
(ESC 0 1 1)(1 1 1 0 0 0 1 1)

FENI: megszakítás engedélyezés
(ESC 0 1 1)(1 1 1 0 0 0 0 0)

FDISI: megszakítás letiltása
(ESC 0 1 1)(1 1 1 0 0 0 0 1)

FLDCW: 8087 vezérlő regiszter betöltése a memóriából
(ESC 0 0 1)(mod 1 0 1 r/m)(LO byte)(HI byte)

FSTCW: 8087 vezérlő regiszter a memóriába
(ESC 0 0 1)(mod 1 1 1 r/m)(LO byte)(HI byte)

FSTSW: 8087 státusz regiszter a memóriába
(ESC 1 0 1)(mod 1 1 1 r/m)(LO byte)(HI byte)

FCLEX: státusz regiszter törlése
(ESC 0 1 1)(1 1 1 0 0 0 1 0)

FSTENV: a 8087 összes regiszterének elmentése a memóriába, kivéve az adat regisztereket
(ESC 0 0 1)(mod 1 1 0 r/m)(LO byte)(HI byte)

FLDNEV: a 8087 regisztereinek betöltése a memóriából (kivéve az adat regisztereket)
(ESC 0 0 1)(mod 1 0 0 r/m)(LO byte)(HI byte)

FSAVE: a 8087 állapotának elmentése
(ESC 1 0 1)(mod 1 1 0 r/m)(LO byte)(HI byte)

FRSTOR: a 8087 állapotának visszatöltése a memóriából
(ESC 1 0 1)(mod 1 0 0 r/m)(LO byte)(HI byte)

FINCSTP: adatregiszter mutató értékének növelése 1-gyel
(ESC 0 0 1)(1 1 1 1 0 1 1 1)

FDECSTP: adatregiszter mutató értékének növelése 1-gyel
(ESC 0 0 1)(1 1 1 1 0 1 1 0)

FFREE: adatregiszter tartalmának törlése
(ESC 1 0 1)(1 1 0 0 0 ST(i))

FNOP: üres utasítás
(ESC 0 0 1)(1 1 0 1 0 0 0 0)

FWAIT: várakozó utasítás
(1 0 0 1 1 0 1 1)

Jelmagyarázat

mod értékei:
00 - nincsenek a címzésben LO HI byte-k
01 - előjeles LO byte van
10 - LO és HI byte van
11 - r/m ST(i)-nek felel meg

r/m értékei:
000 - EA = (BX) + (SI) + byte-k
001 - EA = (BX) + (DI) + byte-k
010 - EA = (BP) + (SI) + byte-k
011 - EA = (BP) + (DI) + byte-k
100 - EA = (SI) + byte-k
101 - EA = (DI) + byte-k
110 - EA = (BP) + byte-k
111 - EA = (BX) + byte-k

mf értékei:
00 - 32 bites valós szám
01 - 32 bites egész szám
10 - 64 bites valós szám
11 - 16 bites egész szám
ST(0) - az aktuális adatregiszter
ST(i) - az i. adatregiszter
d - 0 esetén célregiszter az ST(0),
1 esetén az ST(i).
p - 0 esetén a regiszter nem változik,
1 esetén tartalma elvész.
r - 1 esetén a művelethez tartozó tagok felcserélődnek.

4.4 A 80186 és a 80286 utasításkészlete

A 80186-os és a 80286-os mikroprocesszorok utasításkészlete a 8086/8088 mikroprocesszor teljes utasításkészletét, valamint a következő utasításokat tartalmazza:

BOUND *reg,mem*
Azt kérdezi le, hogy egy érték eleme-e valamely intervallumnak.

ENTER *konst 16,konst 8*
Eljárásba való belépés.

INS *mem*,**DX**
Karaktersorozat beolvasása a DX portról.

INSB *mem*,**DX**
Byte-os karaktersorozat beolvasása a DX portról.

INSW *mem*,**DX**
Szavas karaktersorozat beolvasása a DX portról.

LEAVE
Eljárásból való kilépés.

OUTS DX,*mem*
Byte, szó vagy karaktersorozat kiírása a DX portra.

OUTSB DX,*mem*
Byte-os karaktersorozat kiírása a DX portra.

OUTSW DX,*mem*
Szavas karaktersorozat kiírása a DX portra.

PUSHA
Az összes regiszter elhelyezése a veremben.

POPA
Az összes regiszter kivétele a veremből.
A 286-os mikroprocesszor az előzőekben leírt utsításokon kívül privilégizált utasításokat is tartalmaz. Ezek a következők:

ARPL *mem,reg*
A kívánt privilégium-szint beállítása.

CLTS
A taszk-kapcsolt flag törlése.

LAR *reg,mem*
A hozzáférési jogok betöltése.

LGDT *mem*
A 8-byte-os globális leíró tábla betöltése.

LIDT *mem*
A megszakítás-leíró tábla betöltése.

LLDT *mem*
A lokális leíró tábla betöltése.

LMSW *mem*
A gépi állapotjelző szó betöltése.

LSL *reg,mem*
A szegmenshatár betöltése.

LTR *mem*
A taszk-regiszter betöltése.

SGDT *mem*
A 8-byte-os globális leíró tábla tárolása.

SIDT *mem*
A 8-byte-os megszakítás-leíró tábla tárolása.

SLDT *mem*
A lokális leíró tábla tárolása.

SMSW *mem*
A gépi állapotjelző szó tárolása.

STR *mem*
A taszk-regiszter tárolása.

VERR *mem*
Az olvasásra való hozzáférés ellenőrzése.

VERW *mem*
Az írásra való hozzáférés ellenőrzése.

4.5 A 80287 utasításkészlete

A 287 mikroprocesszor utasításkészlete a 8087 teljes utasításkészletét, valamint a következő utasításokat tartalmazza:

FSETPM
Védett mód beállítása.

FSTSW AX
Az állapotjelző szó tárolása AX-ben WAIT-tel.

FNSTSW AX
Az állapotjelző szó tárolása AX-ben WAIT nélkül.

5. Programozás macro-assembly nyelven

5.1 A macro-assembly programok felépítése

A macro-assembly programok modulokból, a modulok szegmensekből, a szegmensek assembly utasításokból, direktívákból és makró-direktívákból állnak. Egy **modul** azon szegmensek összessége, amelyek a fordítás során egy tárgykód (.OBJ) file-ba kerülnek. A **szegmens** a SEGMENT és az ENDS direktívák közti programrészlet. Az assembly utasítások, a direktívák és a makró-direktívák használatának szabályait ismerteti a következő fejezet.

A program szerkezetét, a modulok egymáshoz viszonyított és abszolút elhelyezkedését a memóriában a programozónak kell meghatározni a direktívák segítségével. Ezek a direktívák a következők:

SEGMENT	szegmens definiálása
ENDS	szegmens vége
END	forrás-file vége
GROUP	szegmens-csoportok
ASSUME	szegmens-regiszterek
ORG	szegmens kezdőcíme
EVEN	szegmens szóhatárra-igazítása
PROC	eljárás definiálása
ENDP	eljárás vége

A direktíváknak lehetnek paraméterei, ezeket a továbbiakban attribútumoknak nevezzük.

5.1.1 A macro-assembly utasítások általános alakja

A makró-assembly utasítások hatására a fordító valamilyen tevékenységet hajt végre, pl. gépi kódú utasításokat vagy adat-byte-okat állít elő. Az utasítások általános alakja:

[*címke:*][*mnemonik*[*operandus,,,*]][*;megjegyzés*]

Az utasítások nem lehetnek többsorosak és hosszuk nem haladhatja meg a 128 karaktert. Tetszőleges pozíción kezdődhetnek a képernyőn. Az utasítás végét a képernyőn és a lemezen való tároláskor a soremelés karakter jelzi. Ez a forrás-file utolsó utasítására is vonatkozik!

Példa:

```
alprog: MOV CX,LENGTH vektor  ; ismeretlen tenyezo
```

5.1.2 Az utasításkészletet beállító direktívák

Az utasításkészletet beállító direktívák a következők:

.8086
.8087

.186
.286c
.286p
.287

Ezek a direktívák engedélyezik az adott mikroprocesszor utasításkészletének használatát. Az utasításkészlet-direktívának a program elején kell állnia, ha használjuk.

A fordító alapértelmezésben a 8086 és a 8087 utasításkészletét fordítja, ezért a .8086 és .8087 direktívákat nem kötelező megadni.A .8086 direktíva a 8086 mikroprocesszor utasításkészletének használatát engedélyezi és letiltja a 80186 és 80286 utasításait. A .8087 direktíva a 8087 lebegőpontos társprocesszor utasításainak használatát engedélyezi és letiltja a 80287 utasításait.

A .186 direktíva engedélyezi a 80186 mikroprocesszor utasításainak fordítását.

A .286c direktíva a 80286 mikroprocesszor nem-privilégizált utasításainak használatát engedélyezi. Ezek az utasítások megegyeznek a 80186 utasításaival. A .286p direktíva kiadása után a 80286 mikroprocesszor privilégizált utasításai is használhatók.

A .287 direktíva hatására a 80287 lebegőpontos koprocesszor utasításait fordítja a makró-assembler.

A mikroprocesszor-család assembly utasításainak leírását a negyedik fejezet tartalmazza.

Amikor a .8087 és .287 direktívát használjuk, a fordító indításánál (ld. VII. fejezet) meg kell adni a /R vagy a /E opciót. A /R opció hatására lebegőpontos utasítás esetén a fordító az adott utasítás kódját generálja, /E opció hatására viszont software interrupt kódot állít elő a megfelelő lebegőpontos emulátorrutin hívásához.

5.1.3 A forrás-modulok és az END direktíva

Minden assembly nyelvű program egy vagy több forrás-modulból áll. *A modulok (forrás-file-ok) egyszerű szöveg-file-ok, ezekből a makró-assembler az utasítások lefordításával tárgymodulokat állít elő.* **Minden forrás-file azonos szerkezetű: nulla vagy több programszegmensből és a file végét jelző END direktívából áll. Az END használata kötelező.** Ezzel a direktívával definiálhatjuk a program belépési pontját vagy kezdőcímét, ha van.

A többi direktíva használata nem kötelező.

Alakja:

END [*kifejezés*]

A kifejezés a **program belépési pontját** határozza meg. A belépési pont az a cím, ahol a program végrehajtása kezdődik. Ha a program egynél több modulból áll, akkor csak egy modulban, a főmodulban lehet belépési pontot definiálni. Ha nem adunk meg belépési pontot, nincs alapfeltételezés.

A makró-assembler az END után álló utasításokat nem veszi figyelembe.

Példa:

```
END kezdet
END
```

5.1.4 A SEGMENT és az ENDS direktíva

A modulokat felépítő szegmensek kijelölésére szolgál.

Alakja:

azonosító SEGMENT [*igazítás*][*kombináció*]['*osztálynév*']
azonosító ENDS

A SEGMENT direktíva a szegmens elejét, az ENDS a szegmens végét jelzi. A programszegmens utasításokból és/vagy adatokból áll, az utasítások és adatok címe egy adott szegmensregiszterhez viszonyítva értelmezhető.

Az "azonosító" a szegmens neve, amely lehet egyedi, vagy megegyezhet a program más szegmensének nevével. Az egyező nevűekből a makró-assembler egyetlen szegmenst állít elő. Azonos nevű szegmensek esetén mindegyik szegmens-definícióban vagy pontosan azonos attributumokat kell megadni, vagy olyanokat, amelyek nem mondanadk ellent egymásnak.

Az "igazítás", a "kombináció" és az "osztálynév" a program betöltését határozza meg.

A szegmensen belül másik szegmens is elhelyezkedhet, de átlapolásuk nem megengedett.

Az "igazítás" a szegmens kezdőcímének értékét befolyásolja. Igazításként a következő kulcsszavak valamelyikét lehet megadni:

BYTE	a szegmens byte-határon kezdődik
WORD	a szegmens szóhatáron kezdődik (1 szó = 2 byte)
PARA	a szegmens paragrafushatáron kezdődik (1 paragrafus = 16 byte)
PAGE	a szegmens laphatáron kezdődik (1 lap = 1024 byte)

Az igazítás alapértelmezése: PARA

A "kombináció" azt határozza meg, hogy az azonos nevű szegmenseket hogyan állítsa össze a rendszer. A következő esetek lehetségesek:

PUBLIC	Az azonos nevű szegmensek egymáshoz fűzésével egyetlen, folytonos szegmenst állít elő. Az új szegmens címei az első ilyen nevű szegmens szegmenscíméhez relatívek. Az összefűzés a szegmensek definiálási sorrendjében történik.
STACK	Az azonos nevű szegmensek egymáshoz fűzésével egyetlen, folytonos szegmenst állít elő. A hozzáfűzés a szegmensek definiálási sorrendjében történik. Az új szegmens címei az SS szegmensregiszterhez relatívek. Az SP (Stack Pointer) regiszterbe egy szegmensen belüli cím kerül. Minden programban legalább egy STACK szegmenst definiálni kell. (Kivétel: ha a programból .COM file-t akarunk létrehozni az EXE2BIN DOS paranccsal.)

COMMON	Úgy helyezi el az azonos nevű szegmenseket, hogy a kezdőcímük azonos legyen. Az így kialakított terület hossza egyenlő a leghosszabb szegmens hosszával. A szegmensek címei ugyanahhoz a báziscímhez relatívek.
MEMORY	Az azonos nevű szegmenseket a memória legnagyobb fizikai szegmensében helyezi el. Ha egynél több MEMORY szegmens van, akkor úgy helyezi el a szegmenseket, mint a COMMON-nál.
AT *cím*	A szegmens címei a megadott "cím"-hez relatívek. A "cím" kifejezés is lehet, de nem tartalmazhat olyan szimbólumra való hivatkozást, amelyet később definiálunk a programban. Az AT szegmensek általában nem kódot és adatokat tartalmaznak, hanem címeket, amelyek segítségével memóriabeli kódok és adatok olyanná tehetők, mintha ROM-ban (Read Only Memory) lennének. Az ilyen utasítások és adatok az AT szegmens címkéivel és változóival érhetők el.

Ha nem adunk meg "kombinációt", akkor betöltéskor maga a fizikai szegmens kerül a memóriába.

Az "osztálynév" megadásával a szegmenst az adott nevű osztályba soroljuk. Az egy osztályba tartozó szegmensek folytonos memóriaterületre töltődnek, közöttük nem helyezkedhet el más osztályhoz tartozó szegmens. Az osztálynevet aposztrófok közé kell tenni. Az osztálynév szintaxisa megegyezik az azonosítók szintaxisával.

5.1.5 Az ORG direktíva

Alakja:

ORG *kifejezés*

Az ORG direktíva hatására a "kifejezés" értéke az utasításszámlálóba kerül. Ez lesz a következő utasítás vagy adat címe az aktuális szegmensen belül. A kifejezés értékének egy konkrét számnak kell lennie, vagyis a kifejezésben szereplő minden szimbólumnak ismertnek kell lennie a fordítás első menetében. A kifejezésben a számlálóra hivatkozó szimbólum ($) is előfordulhat.

(Megjegyzés: ha programunkból .COM file-t akarunk létrehozni, a szegmensben meg kell adni az ORG 100H direktívát.)

5.1.6 Az EVEN direktíva

Alakja:

EVEN

Az EVEN direktíva az azt követő adatot vagy utasítást szóhatárra igazítja. Ha a számláló aktuális értéke páratlan, akkor a direktíva hatására egy NOP (üres) utasítást (1 byte-ot) generál a fordító, így a számláló értéke 1-gyel nő és páros lesz. Ha a számláló értéke páros, akkor az EVEN direktíva hatástalan. Byte-határra igazított szegmensekben nem lehet az EVEN direktívát használni.

5.1.7 A PROC és az ENDP direktíva

Ezt a két direktívát eljárások deklarálására használjuk.

Alakja:

név **PROC** [*típus*]
utasítások
név **ENDP**

A PROC direktíva az eljárás kezdetét, az ENDP direktíva a végét jelzi. Az eljárás nevének egyértelműen kell azonosítania az eljárást. A név lehet JMP, CALL és LOOP utasítás operandusa is. Az eljárás típusát a NEAR vagy a FAR kulcsszóval határozhatjuk meg. Ha a típust nem adjuk meg, akkor NEAR az alapértelmezés.

A PROC és az ENDP direktívák között tetszőleges számú utasítás helyezkedhet el. Minden eljárásnak legalább egy RET utasítást kell tartalmaznia, amelynek hatására visszakerül a vezérlés a hívás helyére.

Az eljárások nem paraméterezhetők.

5.1.8 A GROUP direktíva

A GROUP direktívát szegmenscsoportok kialakítására használjuk.

Alakja:

azonosító GROUP *szegmensnév*,,,

A GROUP direktívával egy vagy több szegmensből álló csoportot definiálhatunk. Az "azonosító" a szegmenscsoport nevét jelenti. A szegmenscsoport címkéinek és változóinak címe a csoport elejéhez relatív, nem annak a szegmensnek a kezdetéhez, amelyben definiálták. A szegmenscsoport nevének egyedinek kell lennie. A "szegmensnév" olyan azonosító, amelyet SEGMENT direktívában vagy SEG kifejezésben definiáltunk.

Az egy szegmenscsoportba tartozó különféle szegmensek az elsőként megadott szegmens szegmenscíméhez relatívek, így a csoport objektumainak címzése egyszerűbb.

A GROUP direktíva nem határozza meg a csoport szegmenseinek betöltési sorrendjét. A betöltési sorrend vagy a szegmensek osztályától függ, vagy attól, hogy a tárgymodulok milyen sorrendben kerülnek a futtatható programot előállító szerkesztőhöz.

A csoport szegmenseinek nem kötelező folytonosan elhelyezkedniük, vagyis a szegmensek közé olyan szegmens is betölthető, amely nem tartozik a csoporthoz. Figyelembe kell azonban venni, hogy a csoport első szegmensének első byte-ja és a csoport utolsó szegmensének utolsó byte-ja közötti távolság nem haladhatja meg a 65535 byte-ot. Ha a csoport szegmensei folyamatosan helyezkednek el, akkor a csoport mérete elérheti a 64 Kbyte-ot.

A csoportneveket ASSUME direktívában (ld. 1.9) és a címkiszámítás viszonyítási alapjának megváltoztatására lehet használni.

A forrás-file-ban egy adott csoportnevet csak egyetlen GROUP direktívában lehet használni. **Ha a csoport több szegmensből áll, akkor a csoporthoz tartozó összes szegmens nevét ugyanabban a GROUP direktívában kell megadni.**

5.1.9 Az ASSUME direktíva

Az ASSUME direktívával adott szegmensekhez adott szegmens-regisztereket rendelünk.

Alakja:

ASSUME *szegmens-regiszter:szegmensnév,,,*
ASSUME NOTHING

Az ASSUME direktíva a "szegmensregiszter"-hez hozzárendeli a "szegmensnév"-vel megadott szegmens szegmenscímét. Ezután a fordító minden címképzési folyamatban az objektumok címét a címképzési módhoz tartozó szegmensregiszterhez rendelt szegmens szegmenscíme alapján számítja ki.

Szegmensregiszterként a CS, DS, ES vagy SS regiszter adható meg.

Az ASSUME direktívával biztosítjuk, hogy a szegmens objektumai elérhetők az adott szegmensregiszterrel. Ennek megvalósulásához azonban előzőleg a programozónak gondoskodnia kell a szegmensregiszter értékének beállításáról is!

A szegmensnév a következők valamelyike lehet:

- SEGMENT direktívával már előzőleg definiált szegmens azonosítója
- GROUP direktívával már előzőleg definiált csoport azonosítója
- a NOTHING kulcsszó.

A direktíva ASSUME NOTHING alakjának hatására az összes előző szegmenskijelölés megszűnik.

A címkiszámítás viszonyítási alapját megváltoztató műveletet (:) felhasználhatjuk az ASSUME direktívával kijelölt szegmens megváltoztatására.

5.1.10 A megjegyzések és a COMMENT direktíva

A megjegyzések tetszőleges karakterekből állhatnak. Bárhol elhelyezhetők a programban. A megjegyzéseket a fordító nem veszi figyelembe. Az egy sornál nem hosszabb megjegyzések alakja:

;szöveg

Az egy sornál hosszabb megjegyzések írására a COMMENT direktívát használjuk.

Alakja:

COMMENT *határoló szöveg határoló*

A COMMENT direktíva hatására a fordító megjegyzésnek tekinti a határoló jelek közötti szöveget. A határoló jel a COMMENT kulcsszó után következő első olyan karakter, amely nem szóköz. A megjegyzés a határoló jel legközelebbi előfordulásáig tart. *A második határoló jel sorában a határoló jel utáni karaktereket sem veszi figyelembe a fordító.*

5.1.11 Példák a makró-assembly programok felépítését meghatározó direktívák használatára

A mintaprogram két modulból áll, az első az 'indító', a második a 'kiíró' modul. Ezeknek természetesen két forrásfile-ban kell lenniük (INDITO.ASM és KIIRO.ASM). Mindegyik modulban több szegmens van, az első modul a főprogram, tehát ebben adjuk meg a program belépési (indítási) pontját (ez a 'start' címke). Indulás után a program beállítja az adatszegmensregisztert, majd a második modulban levő kiírás címkére adja a vezérlést, ami DOS hívás segítségével kiírja a 'szöveg'-et, majd visszaadja a vezérlést az első modul 'vége' címkéjére, amely visszatér a DOS-hoz.

Figyeljük meg, hogy mindkét modulban az elérni kívánt szegmensek deklarációja PUBLIC, a másik modul által használt címkék szintén PUBLIC-ként vannak kijelölve, a másik modulbol hívott címke pedig külső (EXTRN) megjelölést kap. Mindket modul az END direktívával zárul, de csak a főprogramnak szánt 'indító' modulban adjuk meg az indítócímet.

Az INDITO.ASM file listája:

```
        NAME    indito
        PUBLIC  start, vege
        EXTRN   kiiras:near
stack   SEGMENT WORD PUBLIC 'STACK'
        dw     64 DUP(?)
stack   ENDS
data    SEGMENT WORD PUBLIC 'DATA'
data    ENDS
code    SEGMENT BYTE PUBLIC 'CODE'
        ASSUME  cs:code,ds:data
start:  mov     ax, seg DATA
        mov     ds, ax
        jmp     kiiras
vege:   mov     ax, 4C00H
        int     21H
code    ENDS
        END     start
```

A KIIRO.ASM file listája:

```
        NAME    kiiro
        PUBLIC  kiiras
        EXTRN   vege:near
data    SEGMENT WORD PUBLIC 'DATA'
szoveg  db      'Mintaprogram'
        db      13,0AH,'$'
data    ENDS
code    SEGMENT BYTE PUBLIC 'CODE'
        ASSUME  cs:code,ds:data
```

```
kiiras: mov     dx, offset szoveg
        mov     ah, 9
        int     21H
        jmp     vege
code    ENDS
        END
```

A két modulból a következő módon kell futtatható programot készíteni:

1. Fordítsuk le az első modult:
   `C>MASM INDITO,,;`
2. Fordítsuk le a második modult:
   `C>MASM KIIRO,,;`
3. Szerkesszük össze a két modult futtatható programmá:
   `C>LINK INDITO+KIIRO;`

Az 1. lépés hatására egy
INDITO.OBJ és egy
INDITO.LST

a 2. lépés hatására egy
KIIRO.OBJ és egy
KIIRO.LST

a 3. lépés után pedig egy
INDITO.EXE

file keletkezik a C meghajtón. Az LST file-ok a modulok listait, az OBJ-ok a legfordított programot, az EXE file pedig az összeszerkesztett kész programot tartalmazzák.

Ezután írjuk be az

```
C>INDITO<Enter>
```

parancsot és a képernyőn megjelenik a

```
Mintaprogram
```

szöveg, amit programunk írt ki.

5.2 Deklarációk

Ebben a fejezetben azt írjuk le, hogyan lehet adatokat előállítani a program számára, hogyan lehet az utasításokra és adatokra vonatkozó címkéket, változókat és egyéb szimbólumokat deklarálni, valamint hogyan definiálhatók a több mezőből álló struktúrák és rekordok.

5.2.1 Adatdeklarációk

Az adatdeklarációk lehetővé teszik a programozó számára, hogy programjához adatokat definiáljon. A direktívák byte-okra, szavakra vagy egyéb memóriaegységekre képezik le a számokat, karaktersorozatokat és kifejezéseket. Az adatok bekerülnek a program tárgykódú változatába, a program object file-jába.

A következő adatdeklaráló direktívák adhatók meg:

DB	byte
DW	szó (= 2 byte)
DD	dupla szó
DQ	négy szó
DT	10 byte-os összefüggő terület

A DB direktíva (Define Byte)

Alakja:

[*azonosító*] DB *kezdőérték*[,,,]

A DB direktíva konstansok, vagy (kezdőértékkel ellátott) változók deklarálására alkalmas. A konstans vagy a változó kezdőértéke egész szám, karaktersorozat, DUP művelet (ld. ebben a fejezetben később), konstans kifejezés vagy kérdőjel lehet. A kérdőjel azt jelenti, hogy a kezdőérték definiálatlan. Két vagy több kezdőérték megadása esetén az értékeket vesszővel kell elválasztani egymástól. A DB direktíva a karaktersorozat kivételével minden kezdőérték számára 1 byte-ot foglal le. A karaktersorozat kezdőértékhez a direktíva annyi byte-ot rendel, amennyi a sorozat karaktereinek száma.

Az "azonosító" megadása nem kötelező. Ha megadjuk, akkor a DB direktíva egy olyan BYTE típusú változót állít elő, amelynek relatív címe a számláló aktuális értékével egyenlő. (Byte típusú változó: a változóra való hivatkozáskor 1 byte-nyi területre hivatkozunk.)

A karaktersorozat-konstans legfeljebb annyi karakterből állhat, amennyi az adott DB direktíva sorában elfér. A karakterek a felírási sorrendben tárolódnak, vagyis az első karakter címe a legkisebb és az utolsóé a legnagyobb.

Egyetlen DB direktívával különböző típusú kezdőértékek megadhatók, de ekkor a kezdőértékek között nem szerepelhet DUP művelet.

Példák:

```
egész              DB 3
karaktersorozat    DB 'MUNKAHELYE:'
vektor             DB 5 DUP(0)
konstans-kifejezés DB 10*10
definiálatlan      DB ?
vegyes             DB 6, 4*2, 'SZOVEG', ?
```

A DW direktíva (Define Word)

Alakja:

[*azonosító*] DW *kezdőérték*[,,,]

A DW direktíva konstansok vagy (kezdőértékkel ellátott) változók deklarálására alkalmas. A konstans vagy a változó kezdőértéke egész szám, karaktersorozat, DUP művelet, konstans kifejezés, címkifejezés vagy kérdőjel lehet. A kérdőjel definiálatlan kezdőértéket jelent.

Ha az opcionális "azonosító"-t megadjuk, akkor a DW direktíva egy olyan WORD típusú változót állít elő, amelynek relatív címe a számláló aktuális értékével egyenlő. (WORD típusú változó: a változóra való hivatkozáskor 1 szónyi területre hivatkozunk.)

A DW direktíva minden megadott kezdőérték számára egy-egy szót (1 szó = 2 byte) foglal le. A karaktersorozat-konstansok legfeljebb 2 karakter hosszúságúak lehetnek. Az utolsó vagy egyetlen karakter az alacsonyabb helyiértékű byte-on, az első karakter a magasabb helyiértékű byte-on tárolódik. Ha a string egyetlen karakterből áll, akkor a magasabb helyiértékű byte-ra nulla kerül.

Egyetlen DW direktívával különböző típusú kezdőértékek megadhatók, de ekkor a kezdőértékek között nem szerepelhet DUP művelet.

Példák:

```
egész               DW 318
karaktersorozat     DW 'ab'
vektor              DW 5 DUP(0)
konstans-kifejezés  DW 10*7
címkifejezés        DW egész
definiálatlan       DW ?
vegyes              DW 512, 2*3, 'cd', ?
```

A DD direktíva (Define Doubleword)

Alakja:

[*azonosító*] DD *kezdőérték*[,...]

A DD direktíva minden megadott "kezdőérték" számára egy-egy dupla szavas (4 byte-os) tárterületet foglal le, és beállítja az adott kezdőértékre. A "kezdőérték" egész szám, valós szám, karaktersorozat-konstans, kódolt valós szám, DUP művelet, konstans kifejezés, címkifejezés vagy kérdőjel lehet. A kérdőjel definiálatlan kezdőértéket jelent.

Ha az opcionális "azonosító"-t megadjuk, akkor a DD direktíva egy olyan DWORD típusú változót állít elő, amelynek relatív címe a számláló aktuális értékével egyenlő. (DWORD típusú változó: a változóra való hivatkozáskor egy dupla szónyi tárterületre hivatkozunk.)

A karaktersorozat-konstansok legfeljebb 2 karakter hosszúságúak lehetnek. Ha egy karakterünk van, az a legalacsonyabb helyiértékű byte-ba kerül, a többi byte 0-val töltődik fel, ha kettő, akkor abból a másodiknak leírt kerül a legalacsonyabb helyiértékű byte-ra, az elsőnek leírt az ez utáni byte-ra, a többi byte 0 lesz.

Egyetlen DD direktívával különböző típusú kezdőértékek megadhatók, de ekkor a kezdőértékek között nem szerepelhet DUP művelet.

Példák:

```
egész               DD 318
```

```
valós               DD 31.8
karaktersorozat     DD 'ab'
kódolt-valós        DD 3F800000R
vektor              DD 5 DUP(0)
konstans-kifejezés  DD 10*7
címkifejezés        DD egész
definiálatlan       DD ?
vegyes              DD 622, 42.3, 'cd', kódolt-valós, ?
```

A DQ direktíva (Define Quadword)

Alakja:

[*azonosító*] DQ *kezdőérték*[,,,]

A DQ direktíva konstansok vagy (kezdőértékkel ellátott) változók deklarálására alkalmas. A konstans vagy a változó kezdőértéke egész szám, valós szám, karaktersorozat, kódolt valós szám (ld. II. fejezet), DUP művelet, konstans kifejezés vagy kérdőjel lehet. A kérdőjel definiálatlan kezdőértéket jelent.

Ha az opcionális azonosítót megadjuk, akkor a DQ direktíva egy olyan QWORD típusú változót állít elő, amelynek relatív címe a számláló aktuális értékével egyenlő. (QWORD típusú változó: a változóra való hivatkozáskor egy 4 szónyi tárterületre hivatkozunk.)

A DQ direktíva minden megadott kezdőérték számára 4 szót foglal le.

A karaktersorozat-konstansok legfeljebb 2 karakter hosszúságúak lehetnek. Ha egy karakterünk van, az a legalacsonyabb helyiértékű byte-ra kerül, a többi byte 0-val töltődik fel, ha kettő, akkor abból a másodiknak leírt kerül a legalacsonzabb helyiértékű byte-ra, az elsőnek leírt az ez előtti byte-ra, a többi byte 0 lesz.

Egyetlen DQ direktívával különböző típusú kezdőértékek megadhatók, de ekkor a kezdőértékek között nem szerepelhet DUP művelet.

Példák:

```
egész               DQ 318
valós               DQ 31.8
karaktersorozat     DQ 'ab'
kódolt-valós        DQ 3F80000000000000R
vektor              DQ 5 DUP(0)
konstans-kifejezés  DQ 10*7
definiálatlan       DQ ?
vegyes              DQ 512, 44.6, 'cd', 2*3, ?
```

A DT direktíva (Define Tenbytes)

Alakja:

[*azonosító*] DT *kezdőérték*[,,,]

A DT direktíva konstansok vagy (kezdőértékkel ellátott) változók deklarálására alkalmas. A konstans vagy a változó kezdőértéke egész kifejezés, tömörített decimális szám, karaktersorozat-konstans, kódolt valós szám, DUP művelet vagy kérdőjel lehet. A kérdőjel definiálatlan kezdőértéket jelent.

Ha az opcionális azonosítót megadjuk, akkor a DT direktíva egy olyan TBYTE típusú változót állít elő, amelynek relatív címe a számláló aktuális értékével egyenlő. (TBYTE típusú változó: a változóra való hivatkozáskor egy 10 byte-nyi tárterületre hivatkozunk.)

A DT direktíva minden megadott kezdőérték számára egy 10 byte-os területet foglal le.

A karaktersorozat-konstansok legfeljebb 2 karakter hosszúságúak lehetnek. Ha egy karakterünk van, az a legalacsonyabb helyiértékű byte-ra kerül, a többi byte 0-val töltődik fel, ha kettő, akkor abból a másodiknak leírt kerül a legalacsonyabb helyiértékű byte-ra, az elsőnek leírt az ez utáni byte-ra, a többi byte 0 lesz.

A DT direktíva alapfeltételezése, hogy a decimális konstansok tömörített decimálisak, nem egész számok. Ezért a decimális egészek után "D"-t kell tenni.

Egyetlen DT direktívával különböző típusú kezdőértékek megadhatók, de ekkor a kezdőértékek között nem szerepelhet DUP művelet.

Példák:

```
egész                  DT 318D
tömörített-decimális   DT 318
karaktersorozat        DT 'ab'
kódolt-valós           DT 3F800000000000000000R
vektor                 DT 5 DUP(0)
definiálatlan          DT ?
vegyes                 DT 193D, 'cd', ?
```

A DUP művelet

Felírási módja:

szám DUP(*kezdőérték,,,*)

A DUP műveleti jelet olyankor használjuk, amikor egy vagy több kezdőértéket ismételni akarunk. A "szám" határozza meg, hogy hányszor ismétlődjön a kezdőérték. A kezdőérték lehet kérdőjel karakter (határozatlan kezdőértéket jelent), vagy egy olyan kifejezés, amelynek értéke egy egész szám, karakterkonstans vagy egy másik DUP művelet. *A DUP műveletek 17 szint mélységig ágyazhatók egymásba.* Ha egynél több kezdőértéket adunk meg, akkor a kezdőértékeket vesszővel kell elválasztani egymástól. Ilyenkor a kezdőértékek sorozatonként ismétlődnek.

Példák:

```
DB 10 DUP(10 DUP(1))
```

Ez a direktíva 100 byte-ot foglal le és mindegyik byte az 1 értéket tartalmazza.

```
DW 4 DUP(3 DUP(2 DUP(9,8,7)))
```

Ez a direktíva 24 szót foglal le és a szavak rendre a 9,8,7,9,8,7,... értéket tartalmazzák.

DB 20 DUP(?)

A direktíva hatására a fordító 20 byte-ot foglal le és a byte-ok tartalma definiálatlan.

DB 3 DUP(2 DUP(4,6))

Ez a direktíva 6 byte-ot foglal le, amelyek értéke a legkisebb című byte-tól a legnagyobb című byte felé haladva: 4,6,4,6,4,6

5.2.2 Címkedeklarációk

A címkedeklarációval olyan azonosítót definiálhatunk, amellyel egy adott utasítás címére hivatkozhatunk. A címkéket a program vezérlésátadó utasításaiban használhatjuk (JMP, CALL, LOOP utasítás).

A címkék NEAR vagy FAR típusúak lehetnek. A NEAR típusú címkére csak abban a szegmensben hivatkozhatunk, amelyikben a címkét definiáltuk. A FAR típusú címkére a program bármely szegmensében hivatkozhatunk.

NEAR típusú címkéket kétféleképpen deklarálhatunk:

azonosító: [*utasítás*]

vagy

azonosító PROC NEAR

megadásával.

FAR típusú címkét a következőképpen definiálhatunk:

azonosító PROC FAR

Egy adott NEAR típusú címkét egy szegmensen belül csak egyszer, egy FAR típusú címkét pedig egy programon belül csak egyszer definiálhatunk.

A címke a deklarációját követő első utasítás címét jelenti, értéke az utasításszámláló aktuális értékével lesz egyenlő.

A címke a deklarációját tartalmazó szegmenshez kijelölt szegmensregiszterhez (ld. 5. fejezet 1.9 pont ASSUME direktíva) relatív.

Címkét a LABEL és a THIS direktívával is deklarálhatunk, ezek ismertetése e fejezetben található.

5.2.3 Szimbólumdeklarációk

A szimbólumdeklaráló direktívák lehetővé teszik a programozó számára, hogy szimbólumokat definiáljon és használjon. A szimbólum egy olyan azonosító, amelynek értéke egy szám, egy szöveg, egy utasítás vagy egy cím lehet. A szimbólumok használata olvashatóbbá teszi a programot. A szimbólumra bármely olyan helyen hivatkozhatunk, ahol a szimbólumhoz rendelt érték használata megengedett.

A következő szimbólumdeklaráló direktívák adhatók meg:

= szimbólum hozzárendelése egy 16-bites értékhez

EQU	szimbólum hozzárendelése egy 16-bites értékhez, egy karaktersorozat-konstanshoz vagy karakterek egyéb kombinációjához, szimbólumok átnevezése
LABEL	adott típusú változó vagy címke definiálása

Az = direktíva

Alakja:

azonosító = kifejezés

Az = direktíva hozzárendeli az "azonosító"-val jelölt szimbólumhoz a "kifejezés" numerikus értékét. Az azonosítóval 16-bites értékre hivatkozhatunk. A direktíva hatására nem történik helyfoglalás. Az azonosítóra való hivatkozáskor a fordító mindig a kifejezés értékét helyettesíti be a szimbólum helyére.

A "kifejezés" egész szám, 1 vagy 2 karakterből álló karaktersorozat-konstans, konstanskifejezés vagy címkifejezés lehet. A kifejezés értéke nem lehet nagyobb, mint 65535. Az "azonosító" vagy olyan szimbólum, amelyet még nem definiáltunk, vagy olyan, amelyet korábban egy = direktívával deklaráltunk, az = direktívával deklarált szimbólumok ugyanis **újradefiniálhatók**.

Példák:

```
egész              = 8364
karaktersorozat    = 'ab'
konstans-kifejezés = 5*10
címkifejezés       = egész
```

Az EQU direktíva

Alakja:

azonosító EQU *kifejezés*

Az EQU direktíva hozzárendeli az "azonosító"-val jelölt szimbólumhoz a "kifejezést". A szimbólummal egy 16-bites értékre, egy karaktersorozatra vagy karakterek egyéb kombinációjára hivatkozhatunk. Az EQU direktívát szimbólumok átnevezésére is lehet használni.

A fordító az "azonosító" minden további előfordulását egy szöveggel vagy a "kifejezés" értékével helyettesíti, attól függően, hogy milyen típusú az EQU direktívában megadott kifejezés.

Az azonosítónak olyan szimbólumnak kell lennie, amelyet előzőleg még nem definiáltunk. A "kifejezés" egész szám, karaktersorozat-konstans, valós szám, kódolt valós szám, utasítás mnemonikus kódja, konstans kifejezés vagy címkifejezés lehet. Az olyan kifejezéssel, amelynek értéke egy [0,65535] intervallumba eső egész szám, egy 16-bites értéket reprezentáló szimbólumot deklarálhatunk. Ekkor az "azonosító" minden előfordulását a megadott értékkel helyettesíti a fordító. Bármely más kifejezés megadása esetén az "azonosító"-val egy szöveg típusú szimbólumot deklarálunk, így az "azonosító" helyébe az adott szöveg kerül.

Az EQU direktívával deklarált szimbólumokat nem lehet újradefiniálni.

Példák:

```
egész                        EQU 8364
karaktersorozat-konstans     EQU 'üzenet'
valós                        EQU 2.5
mnemonik                     EQU ADD
konstans-kifejezés           EQU 5*10
címkifejezé                  EQU egész
```

A LABEL direktíva

Alakja:

azonosító **LABEL** *típus*

A LABEL direktíva lehetővé teszi, hogy a memória valamely területére különböző típusú azonosítókkal hivatkozhassunk, pl. egy duplaszó (DWORD típus) első byte-jára (BYTE típusként) és így tovább. A LABEL direktíva hatására nem történik helyfoglalás. A fordító hozzárendeli az azonosítóhoz a számláló aktuális értékét és a direktívában megadott típust. *Az azonosítónak olyan szimbólumnak kell lennie, amelyet előzőleg még nem definiáltunk.*

Típusként a következő kulcsszavak valamelyike adható meg:

változók típusa
    BYTE
    WORD
    DWORD
    QWORD
    TBYTE

címkék típusa
    NEAR
    FAR

A FAR típusú utasításcímkéket kötelező LABEL vagy PROC direktívával deklarálni, mert a fordító alapfeltételezése a címkék típusára vonatkozóan NEAR.

Példa:

```
a LABEL   BYTE
b DD ?
```

A programban a "b" változóval egy duplaszóra hivatkozhatunk, az "a" változóval pedig ennek a duplaszónak az első byte-jára.

5.2.4 Struktúrák deklarálása

A struktúrák olyan adatterületek, amelyek több, különböző típusú adatból állnak. Deklarálásuk két lépésből áll: először a STRUC, ENDS direktívapár segítségével a struktúra szerkezetét deklaráljuk, azután a struktúra elemeinek kezdőértékét. Az első lépést **struktúra-típusdeklarációnak**, a másodikat **struktúra-deklarációnak** nevezzük.

Struktúra-típusdeklaráció (a STRUC és az ENDS direktíva)

A struktúra típus deklarálása:

struktúranév STRUC
Mező-definíciók
struktúranév ENDS

A STRUC direktíva a struktúra típusdeklarációjának a kezdetét, az ENDS direktíva pedig a végét jelenti. A "struktúranév" a struktúra-típus neve, amely egyértelműen azonosítja a struktúra-típust. A mező-definíciók a struktúra-típus mezőinek számát, a mezők típusát és alapfeltételezés szerinti értékét határozzák meg. Tetszőleges számú mező-definíció megadható. Egy mező-definíció a következők valamelyike lehet:

[*mezőnév*] DB *alapfeltételezés*,,,
[*mezőnév*] DW *alapfeltételezés*,,,
[*mezőnév*] DD *alapfeltételezés*,,,
[*mezőnév*] DQ *alapfeltételezés*,,,
[*mezőnév*] DT *alapfeltételezés*,,,

Az opcionális "mezőnév" a mező neve, amelynek egyértelműnek kell lennie. A mezőnév az adott mezőnek a struktúra kezdetétől való távolságát reprezentálja. A DB, DW, DD, DQ és DT direktíva a mező méretét határozza meg.

Egy, a programban már definiált struktúra-típus felhasználható struktúra-változó deklarálásához. Ha a struktúra-változó deklarálásakor valamely mezőhöz nem adunk meg kezdőértéket, akkor a mező értéke a struktúra-típus deklarációban megadott alapfeltételezéssel lesz egyenlő.

Az "alapfeltételezés" egy szám, karakter- vagy karaktersorozat-konstans vagy szimbólum lehet és DUP műveletet is tartalmazhhat. Ha az alapfeltételezést karaktersorozat-konstanssal adjuk meg, akkor a mező annyi byte hosszúságú lesz, ahány karakterből áll a string. A struktúra-típus deklarációja kizárólag mező-definíciókat és megjegyzéseket tartalmazhat, vagyis a struktúrák nem ágyazhatók egymásba.

Példa:

```
s       STRUC
mező1   DB 'string'
mező2   DB 5
mező3   DW 3 DUP(?)
s       ENDS
```

Struktúra-deklaráció

A struktúra-deklarációk segítségével egy előzőleg már definiált struktúra-típus mezőinek kezdőértékeket adhatunk. A struktúra-deklarációk alakja:

[*azonosító*] *struktúranév* < [*kezdőérték*],,, >

Az "*azonosító*" egy struktúraváltozó neve. A struktúraváltozó egy vagy több mezőből állhat. A különböző mezők különböző méretűek lehetnek. Az azonosítót nem kötelező megadni. Ha elhagyjuk, akkor a fordító területet foglal le a struktúra számára, de nem állít elő olyan azonosítót, amelynek segítségével hozzá lehetne férni a struktúrához.

A "*struktúranév*" egy korábban STRUC direktívával definiált struktúra típus azonosítója.

A "*kezdőértékek*" a struktúra elemeinek kezdőértékét határozzák meg. Minden mezőhöz legfeljebb egy kezdőérték adható meg. A kezdőérték egész szám, karaktersorozat-konstans vagy olyan kifejezés lehet, amely értékének típusa megegyezik a megfelelő mező típusával. A csúcsos zárójelpár (< >) használata akkor is kötelező, ha egyetlen kezdőértéket sem adunk meg. Ha egynél több kezdőérték szerepel a deklarációban, akkor a kezdőértékeket vesszővel kell elválasztani egymástól. A DUP művelet használata esetén csak a kerek zárójelek közötti értékeket kell csúcsos zárójelek közé tenni.

A kezdőértékek megadása nem kötelező. Ha valamely struktúramező számára nem deklarálunk kezdőértéket, akkor a fordító automatikusan a struktúra típus definiálásakor (STRUC direktíva) megadott alapfeltételezés szerinti értéket veszi figyelembe. Ha nincs alapfeltételezés szerinti érték sem, akkor a mező értéke definiálatlan.

A struktúra-deklarációban nem lehet kezdőértéket adni az olyan mezőknek, amelyekhez a STRUC direktívában több alapfeltételezés szerinti értéket rendeltünk.

Példák:

```
str1 s < >
```

A "str1" struktúraváltozót állítja elő, amelynek típusát az "s" struktúra típussal deklaráltuk. A struktúramezők kezdőértékei megegyeznek a struktúra típus deklarációjában megadott alapfeltételezés szerinti értékekkel, ha ezek léteznek.

```
str2 s <,2,>
```

A "str2" struktúraváltozót állítja elő, amely "s" típusú. Az első és a harmadik mező kezdőértéke megegyezik a struktúra típus deklarációjában megadott alapfeltételezés szerinti értékkel. A második mező kezdőértéke 2-re állítódik be.

```
str3 s 5 DUP(<,2,>)
```

A "str3" struktúraváltozót állítja elő, amely 5 darab "s" típusú struktúrából áll. Minden ilyen struktúra első és harmadik mezőjének kezdőértéke megegyezik a struktúra típus deklarálásakor megadott alapfeltételezés szerinti értékkel, és minden ilyen struktúra második mezőjének kezdőértéke 2-re állítódik be.

Példa a struktúra típusdeklaráció és a struktúra-deklaráció együttes használatára:

```
struk STRUC
a     DB ?
b     DB 'darab'
c     DB 'ára (\)'
d     DW ?
struk ENDS
struktúra struk <23,,,2300>
```

A STRUC, ENDS direktívapárral és a köztük lévő DB, DW direktívákkal a "struk" struktúra szerkezetét és a struktúra-elemek alapfeltételezés szerinti kezdőértékét deklaráltuk. A struktúra-típus deklaráció után a "struktúra" struktúra-változó deklarációja következik, amely "struk" típusú struktúra. Az első struktúra-elem kezdőértékét 23-ra állítottuk be, a második és harmadik mező kezdőértéke megegyezik a struktúra típusdeklarációjában megadott megfelelő kezdőértékkel, a negyedik mező kezdőértéke 2300.

5.2.5 Rekordok deklarálása

A rekord olyan egy byte vagy egy szó nagyságú terület, amely egy vagy több bitmezőből áll. Deklarálásuk két lépésből áll: először a RECORD direktívával a rekord szerkezetét deklaráljuk, azután a rekord bitmezőinek kezdőértékét. Az első lépést rekord-típus deklarációnak, a másodikat rekord-deklarációnak nevezzük.

Rekord-típus deklaráció (a RECORD direktíva)

A rekord típus definiálása:

rekordnév **RECORD** *mezőnév:szélesség*[=*kifejezés*][,,,]

A RECORD direktívával egy 8- vagy 16-bites rekordot definiálhatunk, amely egy vagy több mezőből áll.

A "*rekordnév*" annak a rekordtípusnak az azonosítója, amelyet rekord előállításakor használunk. A "*mezőnév*" egy rekordbeli mező azonosítója, amelynek bitekben mért hosszát a "*szélesség*" határozza meg. A "*kifejezés*" a mező alapfeltételezés szerinti értéke. **Ha több mezőt adunk meg, ügyeljünk arra, hogy a mezőszélességek összege ne haladja meg a 16-ot!**

A szélességnek olyan konstansnak kell lennie, amelyre: 1< = szélesség < = 16. Ha a deklarált mezők szélességeinek összege meghaladja a 8 bitet, akkor a rekord 2 byte hosszúságú, egyébként 1 byte hosszúságú lesz.

Ha a mező legalább 7-bites, akkor alapfeltételezés szerinti értékként egy ASCII karakter is megadható.

A kifejezésben csak már definiált adatokra lehet hivatkozni.

A bitmezők elhelyezése a rekordban a felsorolás sorrendjében balról jobbra történik. Ha a deklarált mezők nem töltik ki teljesen a 8 vagy a 16 bitet, akkor az egész rekord jobbra tolódik úgy, hogy az utolsó mező utolsó bitje lesz a rekord legalacsonyabb helyiértékű bitje. Ilyenkor a rekord legmagasabb helyiértékű bitjei egyik mezőhöz sem tartoznak.

Példa:

```
bitmezők RECORD x:2, y:4 = 1, z:7 = 'A'
```

Rekord-deklarációk

A rekord-deklarációk alakja:

[*azonosító*] *rekordnév* < [*kezdőérték*],,, >

A rekordváltozó egy 8- vagy 16-bites érték, amely egy vagy több bitmezőből áll. Az "*azonosító*" a rekordváltozó neve, amelyet nem kötelező megadni. Ha elhagyjuk, akkor a fordító területet allokál a rekord számára, de nem állít elő olyan azonosítót, amellyel a rekordot el lehetne érni.

A "*rekordnév*" egy korábban RECORD direktívával megadott rekord típus azonosítója.

A "*kezdőértékek*" a rekord mezőinek kezdőértékét határozzák meg. Minden mezőhöz legfeljebb egy kezdőérték adható meg. A kezdőérték egész szám, karaktersorozat-konstans vagy olyan kifejezés lehet, amelynek értéke ábrázolható az adott hosszúságú mezőben. A csúcsos zárójelek használata akkor is kötelező, ha egyetlen kezdőértéket sem adunk meg. Ha egynél több kezdőérték szerepel a deklarációban, akkor a kezdőértékeket vesszővel kell elválasztani egymástól.

A DUP művelet használata esetén csak a kerek zárójelek közötti értékeket kell csúcsos zárójelek közé tenni.

A kezdőértékek megadása nem kötelező. Ha valamely rekordmező számára nem deklarálunk kezdőértéket, akkor a fordító automatikusan a rekord típus definiálásakor (RECORD direktíva) megadott alapfeltételezés szerinti értéket veszi figyelembe. Ha nincs alapfeltételezés szerinti érték sem, akkor a mező értéke definiálatlan.

Példák:

r1 bitmezők < >

Az "r1" rekordváltozót állítja elő, amelynek típusát a "bitmezők" rekord-típussal deklaráltuk. A rekordmezők kezdőértékei megegyeznek a rekord típus deklarációjában megadott alapfeltételezés szerinti értékekkel, ha ezek léteznek.

r2 bitmezők <,2,>

Az "r2" rekordváltozót állítja elő, amely "bitmező" típusú. Az első és a harmadik mező kezdőértéke megegyezik a rekord-típus deklarációjaban megadott alapfeltételezés szerinti értékkel. A második mező kezdőértéke 2-re állítódik be.

r3 bitmezők 5 DUP(<,2,'B'>)

Az "r3" rekordváltozót állítja elő, amely 5 darab "bitmezők" típusú rekordot tartalmaz. Minden rekord első mezőjének kezdőértéke megegyezik a rekord-típus deklarációjában megadott alapfeltételezés szerinti értékkel. A rekordok második mezőjének kezdőértéke 2-re, a harmadiké 'B'-re állítódik be.

Példa a rekord-típus deklaráció és a rekord-deklaráció együttes használtára:

rek RECORD bm1:5=8, bm2:4, bm3:7='B'
rekord rek <3,5,>

A RECORD direktívával a "rek" rekord szerkezetét és a rekord bitmezőinek alapfeltételezés szerinti kezdőértékét deklaráltuk. A "bm1" bitmező 5-bites, a "bm2" 4-bites, "bm3" 7-bites. "bm1" alapfeltételezés szerinti kezdőértéke 8, "bm2"-nek nincs alapfeltételezés szerinti kezdőértéke, a "bm3" alapfeltételezés szerinti kezdőértéke a 'B' karakter.

A rekord típusdeklaráció után a "rekord" rekordváltozó deklarációja következik, amellyel az első bitmező értékét 3-ra, a második bitmező értékét 5-re állítjuk be. A harmadik bitmező értéke továbbra is a 'B' karakter marad.

5.2.6 Előre-hivatkozások

A macro-assembler megengedi, hogy cimkére, változóra és egyéb szimbólumra a deklaráló utasítást megelőzve (előre) hivatkozzunk.

Például:

```
        JMP cimke
        ADD ...
        .
        .
        .
cimke:
```

Az előre-hivatkozások helytelen használata fordítási hibákhoz vezethet. Ha a fordító definiálatlan azonosítót talál az első fordítási menetben, akkor előre hivatkozást feltételez. Ha csak az azonosítót adjuk meg, akkor a fordító az alapfeltételezés szerinti típus és szegmensregiszter alapján generálja az utasítás kódját és adatait. Az alapfeltételezés attól függ, hogy milyen utasításban használjuk az előre-hivatkozást. Vezérlésátadó utasításban az alapfeltételezés NEAR típusú címke. Egyéb utasításokban a fordító olyan változót feltételez, amely a DS szegmensregiszterhez relatív. Abban az esetben, ha az alapfeltételezés szerinti típus és szegmensregiszter nem felel meg az azonosító programbeli használatának, fordítási hiba keletkezik.

Az ilyen hibákat a címkiszámítás viszonyítási alapjának megváltoztatásával (:), valamint a PTR és SHORT művelet használatával kerülhetjük el. Ha egy olyan változóra hivatkozunk előre, amely az ES, SS vagy CS regiszterhez relatív, akkor a címkiszámítás viszonyítási alapját megváltoztató művelettel (:) kell meghatározni azt a szegmensregisztert, szegmenset vagy csoportot, amelyhez a változó relatív.

Például a

```
        MOV AX,SS:azonosító
```

hatására az AX regiszterbe az SS szegmensregiszterhez relatív azonosítóval jelölt változó tartalma kerül.

A címkiszámítás viszonyítási alapjának megváltoztatása nélkül a fordító azt feltételezi, hogy a változó a DS regiszterhez relatív.

Ha egy JMP (feltétel nélküli ugrás) utasításban olyan utasítás címkéjére hivatkozunk előre, amely a hivatkozás helyétől 128 byte-nál kisebb távolságra van, akkor célszerű a SHORT műveletet használni. A SHORT művelet elhagyása nem okoz ugyan hibát, de ekkor a fordító azt feltételezi, hogy az utasítás, amelynek a címkéjére előre-hivatkoztunk, 128 byte-nál nagyobb távolságra van. Ilyenkor a fordító egy NOP utasítást is generál, amely 128 byte-nál kisebb távolság esetén felesleges.

Példa:

```
JMP SHORT címke
```

A SHORT művelet azt jelzi, hogy a későbbiekben deklarált "címke" a JMP utasítás helyétől 128 byte-nál kisebb távolságra van.

Ha CALL vagy JMP utasításban előre hivatkozunk egy utasítás címkéjére, akkor a PTR művelettel meghatározhatjuk a címke típusát (NEAR vagy FAR). A fordító alapfeltételezése a NEAR típus, így ezt a típust nem kötelező PTR művelettel megadni. Ha a címke FAR típusú és nem használjuk a PTR műveletet, akkor fázishiba keletkezik. A fázishiba azt jelenti, hogy az adott címke típusa nem azonos a fordítás első és második menetében (NEAR típusúként használtuk a későbbiekben FAR típusúként deklarált címkét).

Példa a címke típusának meghatározására:

```
CALL FAR PTR számol
```

A PTR művelet hatására a fordító FAR típusúnak tekinti a későbbiekben deklarált "számól" címkét. Ha egy szegmensnévre úgy hivatkozunk előre, hogy a programban először egy címkiszámítás viszonyítási alapjaként szerepel, akkor előbb a GROUP direktívával rendeljük hozzá a szegmensnevet egy csoportnévhez, majd az ASSUME direktívával rendeljük egy szegmensregiszterhez a csoportnevet! Ha ezt nem tesszük meg, akkor a fordító ignorálhatja a címkiszámítás viszonyítási alapját megváltoztató műveletet és az alapfeltételezés szerinti szegmensregisztert használja, ami általában hibát okoz a fordítás második menetében.

Példa a helyes használatra:

```
csoport   GROUP verem
          ASSUME SS:csoport
prog      SEGMENT
          .
          .
          .
          ADD AX, verem:elem
          .
          .
          .
```

5.2.7 Globális deklarációk

A lokális címkékre, változókra és egyéb szimbólumokra csak abban a modulban lehet hivatkozni, amelyikben a deklarációjuk szerepel. A globális deklarációk direktívái lehetővé teszik a programozó számára, hogy olyan cimkéket, változókat és egyéb szimbólumokat definiáljon, amelyek a program bármely moduljából elérhetők. Az ilyen szimbólumokat **globális szimbólumoknak** nevezzük. A globális deklarációk direktívái:

PUBLIC
EXTRN

A PUBLIC direktíva hatására az adott modulban lokálisan definiált szimbólumok globálisakká válnak, azaz elérhetőek lesznek a program bármely más moduljából is. A globális szimbólumot a program pontosan egy moduljában kell deklarálni, és e modulban adjuk meg azt is a PUBLIC direktívával, hogy a szimbólum globális. Ezután egy másik modulban a szimbólumot úgy tesszük ismertté, hogy itt viszont külső (external) szimbólumként adjuk meg ay EXTRN direktívával. A globális szimbólum tetszőleges számú modul az EXTRN direktívájában előfordulhat.

A PUBLIC direktíva alakja:

PUBLIC *azonosító*[,,,]

Az "*azonosító*" egy olyan változó, címke vagy abszolút szimbólum azonosítója, amelyet a PUBLIC direktívát tartalmazó modulban deklaráltunk. A PUBLIC direktívában megadott szimbólum csak olyan lehet, amelynek értéke 1 vagy 2 byte-os egész szám vagy string.

Az azonosítóban szereplő kisbetűket a fordító nagybetűkké konvertálja mielőtt az azonosítót az object file-ba másolja. Ha az object file-ban is meg akarjuk őrizni a kisbetűket, akkor a fordító hívásakor meg kell adnia /ML és /MX opciót. Erről az V. fejezet 3. pontjában található bővebb információ.

Az EXTRN direktíva alakja:

EXTRN *azonosító:típus*[,,,]

Az "azonosító" egy olyan változó, címke vagy szimbólum azonosítója, amelyet a program egy másik moduljában deklaráltunk, és ott PUBLIC direktívában is szerepel. Az EXTRN direktívában megadott típusnak meg kell egyeznie azzal a típussal, amelyet abban a modulban adtunk meg, ahol a szimbólumot deklaráltuk. A "*típus*" a következő kulcsszavak valamelyikével adhatók meg:

BYTE	WORD
DWORD	QWORD
TBYTE	
NEAR	FAR
ABS	

Az ABS típus azt jelenti, hogy a szimbólum értéke egy konkrét szám.

Mivel az aktuális cím csak a program összeszerkesztésekor alakul ki, minden, az EXTRN direktívával megadott, azaz külső szimbólumról a fordító feltételezi, hogy a program valamely moduljában PUBLIC-ként is deklaráltuk. A fordító egy alapfeltételezést állít fel arról, hogy ez a szimbólum a program melyik szegmensében helyezkedik el.

Ez az alapfeltételezés attól függ, hogy az EXTRN direktíva hol helyezkedik el a modulban. Ha a direktíva egy szegmens belsejében található, akkor a fordító azt feltételezi, hogy a külsőként megadott azonosító az EXTRN direktívát tartalmazó szegmens szegmensregiszteréhez relatív. Ebben az esetben a szimbólum EXTRN és PUBLIC deklarációjának azonos nevű és azonos attributumokkal rendelkező szegmensekben kell szerepelnie. Ha a direktíva helye minden szegmensen kívülre esik, akkor nincs alapfeltételezés. Ekkor az elem PUBLIC deklarációja a program bármely moduljának bármely szegmensében előfordulhat.

A külső elem alapfeltételezés szerinti szegmensének megváltoztatására a címkiszámítás viszonyítási alapját megváltoztató műveletet (:) használhatjuk.

5.3 Makrok definiálása és hívása

A makró-definíciót úgynevezett makró-direktívák és assembly utasítások segítségével építjük fel. A macro-assembler a következő makró-direktívák használatát engedi meg:

MACRO
LOCAL
PURGE
REPT
IRP
IRPC
EXITM
ENDM

A MACRO direktívával a programozó egy adott nevű makrót definiál, amely legalább egy forrás-nyelvű utasításból áll. A programban, azaz a forrás-file-ban a fordító a makró minden előfordulását, azaz a makró hívásait automatikusan helyettesíti a makró-definícióban megadott utasítás-sorozattal.

A LOCAL direktíva lehetővé teszi, hogy a makróban egyedi címkéket definiáljunk. A PURGE direktíva a makró-definíció törlésére használható. A REPT, IRP és IRPC direktívák segítségével adott utasítás-sorozatok folyamatosan ismételhetők. Az ismétlési tényezőt egy kifejezés értéke, egy paraméterlista elemeinek száma vagy egy sztringben lévő karakterek száma határozza meg.

A makró-direktívában speciális műveleti jeleket lehet használni, amelyek a következők:

&	a helyettesítés jele
;;	a makró-megjegyzés jele
!	a karakter-konstans jele
< >	a szöveg-konstans jele
%	a kifejezés jele

Ezeknek a műveleti jeleknek a makró-definícióban való használatával speciális vezérlési műveleteket lehet végrehajtani.

5.3.1 A MACRO és az ENDM direktíva

A MACRO direktíva a makró-definíció kezdetét, az ENDM direktíva a makró-definíció végét jelzi. A makró-definíciók általános alakja:

```
makrónév    MACRO [formális paraméter[,...]]
            utasítás
            .
            .
            .
            ENDM
```

A *makrónév*nek egyedinek kell lennie. Ezt a nevet használjuk a forrás-file-ban a makró hívására. A formális paraméterek olyan azonosítók, amelyek a makró hívásakor kapnak értéket. Tetszőleges számú formális paraméter adható meg, de el kell férniük a MACRO direktíva sorában.

A makró törzse tetszőleges számú makró-assembly utasításból állhat, amelyekben a formális paraméterek akárhányszor előfordulhatnak. A makró-törzsben MACRO direktíva megadása is megengedett, tehát a makró-definíciók egymásba ágyazhatók.

A makró hívása annyiszor történik meg, ahányszor a makrónévre hivatkozunk a modulban. A megjegyzésekben előforduló makróneveket a fordító nem veszi figyelembe. A makróhívás helyére bemásolja a makró törzsében megadott utasításokat, úgy, hogy a formális paramétereket a híváskor megadott értékekkel, az úgynevezett aktuális paraméterekkel helyettesíti. Az aktuális paramétereket a makró-definícióban felírt formális paraméter sorrend szerint kell megadni: pl. a makró-definícióban másodikként megadott formális paraméter a makró-hívás második aktuális paraméterével helyettesítődik.

A makró-definícióba ágyazott másik makró-definíciót addig nem dolgozza fel a fordítóprogram, amíg a külső makró hívása meg nem történik. Ez azt jelenti, hogy egy makróba ágyazott makrót addig nem lehet hívni, ameddig a programban legalább egyszer nem hívtuk a külső makrót. A makró-definíciók tetszőleges mélységben egymásba skatulyázhatók. A mélységet csak a modul fordításakor rendelkezésre álló memóriaterület mérete korlátozza. A makró-definíció más makrók hívását is tartalmazhatja. *A makrók nem lehetnek rekurzívek, azaz törzsükben nem hívhatók mégegyszer.* Makrókifejtéskor a fordítóprogram a formális paramétereket mindig helyettesíti a makróhívásban megadott aktuális paraméterekkel. Ezért nem célszerű formális paraméterként például regiszternevet használni, mert a makró-definícióban lévő olyan utasítások, amelyek erre a regiszterre hivatkoznak, a makró kifejtésekor helytelenül kerülnének a programba.

A makró utasításait csak a makró hívásakor fordítja le a makró-assembler és csak azon a helyen, ahol az utasításokat a forrás-file-ba illeszti. Ebből következik, hogy a lefordított kód címei a hívás helyéhez, és nem a definíció helyéhez relatívek. *Maga a makró-definíció sohasem fordítódik le.*

Abban az esetben, ha a MACRO direktívát közvetlenül a TITLE, SUBTTL vagy NAME direktíva után használjuk, a fordító TITLE, SUBTTL vagy NAME nevű makrót kezd generálni, így megváltozhat ezen direktívák eredeti hatása. (A TITLE direktívával program listázásakor kiírandó cím szövegét, a SUBTTL direktívával az alcímet, a NAME direktívával a szerkesztő hibajelzéseiben kiírandó modulnevet lehet meghatározni.) Ha ezt a problémát el akarjuk kerülni, akkor a címben, alcímben vagy modulnévben szereplő MACRO szót meg kell változtatni, például kiegészíthetjük egy speciális karakterrel: "-MACRO". (A TITLE, SUBTTL és NAME direktívával kapcsolatban e fejezet 4. pontjában található bővebb információ.)

Makródefinícióval más direktívák és assembly utasítások működése is megváltoztatható.

Példa makró definiálására:

```
kivon MACRO a,b,c
      MOV  AX,a
```

```
SUB  AX,b
MOV  c,AX
ENDM
```

5.3.2 Makróhívások

A már definiált makrók hívása a következő formában történik:

makrónév [*aktuális_paraméter*[,,,]]

A makróhívás hatására a fordító bemásolja a hívás helyére a makrónévvel azonosított makró definíciójában szereplő utasításokat és az utasításokban lévő formális paramétereket a hívásban megadott aktuális paraméterekkel helyettesíti. Aktuális paraméterként azonosítót, konstanst vagy értéklistát lehet megadni. Az aktuális paraméterek száma tetszőleges lehet, de nem folytathatók új sorban.

A fordító az első formális paramétert az első aktuális paraméterrel helyettesíti, a másodikat a másodikkal és így tovább. *Ha a makróhívásban több aktuális paramétert adtunk meg, mint a makró-definícióban lévő formális paraméterek száma, akkor a fordító a felesleges aktuális paramétereket nem veszi figyelembe. Ha a makróhívásban megadott aktuális paraméterek száma kevesebb, mint a definícióban lévő formális paraméterek száma, akkor a makró kifejtése során azok a formális paraméterek, amelyeknek nem feleltethetők meg aktuális paraméterek, kimaradnak az utasításokból, semmivel sem helyettesítődnek.*

Abban az esetben, **ha egy aktuális paraméterként egy értéklistát akarunk megadni, a listát csúcsos zárójelek (< >) közé kell tenni.** A lista elemeit vesszővel kell elválasztani egymástól.

Értéklista aktuális paraméterként való megadásának akkor van értelme, ha a hívott makró törzsében olyan IRP ismétlő direktíva is szerepel, amelynek helyettesítő paramétere azonos az őt tartalmazó makródefiníció valamelyik formális paraméterével.

(Az IRP olyan ismétlő direktíva, amelyben az ismétlések számát a helyettesítő paraméterek száma határozza meg. A direktíva részletes leírása e fejezet 3.10 pontjában található.)

Példa makróhívásra:

```
kivon 10,csökk,marad
```

Három aktuális paraméterrel hívja a "kivon" makrót, amelynek az előző pontban megadott definíciója szerint a következő utasítás-sorozat generálódik:

```
MOV AX,10
SUB AX,csökk
MOV marad,AX
```

5.3.3 Részkaraktersorozat megadása makró-paraméterrel

Valamely karaktersorozat egy részét úgy is megadhatjuk, hogy a helyettesítés jelét (&) és a makró valamely formális paraméterét használjuk. A helyettesítés jelét bárhol lehet alkalmazni, ahol a formális paraméter közvetlenül egy másik karakter előtt vagy után áll, vagy ha a formális paraméter egy olyan karaktersorozatban helyezkedik el, amely aposztrófok között áll.

A helyettesítés jele: &

Megadása:

karaktersorozat&formális_paraméter

vagy

formális_paraméter&karaktersorozat

Példa a helyettesítés jelének használatára:

```
peld  MACRO x
abx   MOV  bx,'&x'
ab&x  JMP  cimke
      ENDM
```

Ha az előzőleg definiált makrót a következőképpen hívjuk:

```
peld a
```

akkor a fordító a következő két utasítást generálja:

```
abx   MOV bx,'a'
aba   JMP cimke
```

Egymásba ágyazott makrók esetében a formális paraméter szimbólumban vagy karaktersorozat-konstansban előforduló, aktuális paraméterrel való helyettesítésekor annyi "&" jelet kell használni, amennyi az ágyazási mélység.

Tekintsük például a következő makró-definíciót:

```
mezőkMACRO p
      IRP  q, < 1,2,3 >  ;;q az IRP direktíva
p&&qDB  q              ;; formális paramétere
      ENDM
      ENDM
```

(Az első & a "p"-hez, a második a "q"-hoz tartozik.)

A makró hívásakor a "p" formális paraméter azonnal helyettesítődik. A "q" formális paraméter azonban csak akkor helyettesítődik, amikor a fordító feldolgozza az IRP direktívát. (Az IRP direktíva részletes ismertetése e fejezet 3.10 pontjában található. Hatása: egy utasítás-sorozatot annyiszor ismétel, amennyi az IRP direktívában megadott helyettesítő paraméterek száma.) A "q" formális paraméter az IRP paraméterlistájában megadott értékek szerint helyettesítődik. Ha a makró hívása például:

```
mezők mez
```

akkor a fordítóprogram a következő sorokat generálja:

```
mez1 DB 1
mez2 DB 2
mez3 DB 3
```

5.3.4 A szöveg-konstans jele

A szöveg-konstans jelének hatására a fordító-program literálnak tekinti a szöveget. A szöveg-konstans megadási módja:

<szöveg>

A szöveg-konstans leggyakrabban makró-hívásokban és az IRP direktívában használjuk abból a célból, hogy a fordító a paraméterlista elemeit egyetlen paraméternek tekintse.

A szöveg-konstans jelét arra is felhasználhatjuk, hogy a fordító literálként értelmezzen speciális karaktereket. Például

<;>

nem azt jelenti, hogy megjegyzés következik, hanem a ";" karaktert.

A szöveg-konstans legfeljebb a sor végéig tarthat. A fordítóprogram mindig eltávolítja a csúcsos zárójelpárt, valahányszor a paraméter előfordul a makróban. Ha egymásba ágyazott makrókat használunk, akkor annyi csúcsos zárójelpárt kell megadni, amennyi az ágyazási mélység.

5.3.5 A karakter-konstans jele

A karakter-konstans megadása:

!karakter

Ennek hatására a fordítóprogram literálként értelmezi a karaktert. Akárcsak a szöveg-konstans jelét, a karakter-konstans jelét is használhatjuk arra, hogy a fordító a speciális karaktereket literálnak tekintse. Például

!; és <;>

azonos hatású.

5.3.6 A kifejezés jele

A kifejezés megadási módja:

%szöveg

A kifejezés jelének hatására a makró-assembler a szöveget kifejezésnek tekinti. Az aktuális számrendszert használva kiszámítja a kifejezés értékét és a szöveget ezzel az értékkel helyettesíti. A "szöveg" helyes assembly nyelvű kifejezés kell, hogy legyen.

A kifejezés jelét olyan makró-hívásokban használjuk, amelyekben nem a kifejezés szövegét, hanem a kifejezés kiszámított értékét akarjuk átadni a makrónak.

A kifejezés legfeljebb a sor végéig tarthat.

Definiáljunk egy olyan makrót, amely képernyőre ír egy szöveget és egy értéket!

```
kiír  MACRO szöveg, érték
      %OUT  * szöveg, érték *
      ENDM
```

Ha például

```
e1 EQU 10
e2 EQU 5
```

akkor a

```
kiír <e1-e2= >,%(e1-e2)
```

makróhívás hatására a "szöveg" formális paraméternek az "e1-e2 =" szöveg-konstans aktuális paraméter felel meg, az "érték" formális paraméter helyére pedig az 5 numerikus érték helyettesítődik be.

5.3.7 A PURGE direktíva - makródefiníciók törlése

A PURGE direktívával adott nevű makrók definícióját törölhetjük a memóriából. A törölt makrók további hívása hibajelzést okoz.

A PURGE direktíva alakja:

PURGE *makrónév*[,,,]

Legfeljebb annyi makrónév adható meg, amennyi a direktíva sorában elfér.

A direktíva csak akkora területet töröl, amekkora helyet a makró-definíció elfoglalt. Ha "makrónév" megegyezik valamely utasítás vagy direktíva mnemonikus kódjával, akkor a PURGE direktíva visszaállítja a mnemonikus kód eredeti jelentését.

A makró-könyvtár egy olyan file, amely makró-definíciókat tartalmaz. A makró-könyvtárat az INCLUDE direktívával lehet a forrás-file-ba illeszteni, és ezután a PURGE direktívával törölhetjük a felesleges makró-definíciókat.

Ha egy makrót újra definiálunk, akkor a makró előző definíciója automatikusan törlődik.

Példa:

```
PURGE makro1,makro2
```

5.3.8 A LOCAL direktíva - egyedi azonosítók létrehozása makrókban

A LOCAL direktíva hatására a makró egyes kifejtéseire nézve egyedi azonosítók generálódnak.

Alakja:

LOCAL *dummy-azonosító*[,,,]

A "*dummy-azonosító*" a makró minden újabb kifejtésekor új egyedi névvel helyettesítődik. A "*dummy-azonosító*" a makró bármely utasításában előfordulhat. A kifejtéskor

??xxxx

alakú egyedi nevet helyettesít be a fordító, ahol xxxx egy [0000,FFFF] intervallumbeli hexadecimális szám.

A LOCAL direktívát rendszerint arra használjuk, hogy a makrókban egyedi címkéket állítsunk elő. Ha ugyanis nem adunk meg LOCAL direktívát, és a makrót többször hívjuk, a kifejtés eredményeképpen többszörösen definiált címkéket kapunk, amelyekre a makró-assembler hibajelzést ad. Ezt a problémát úgy lehet elkerülni, hogy a makró-definícióban szereplő összes címkét LOCAL direktívával adjuk meg, így a makró minden egyes kifejtésekor új, egyedi címkék generálódnak.

A LOCAL direktívát kizárólag makró-definícióban lehet használni és a definíció összes többi utasítását meg kell előznie.

Példa:

```
mac   MACRO a,b
      LOCAL címke
      MOV  AX,b
címke MOV  CX,a
      INC  AX
      JNZ  címke
      ENDM
```

5.3.9 A REPT és az ENDM direktíva - utasításcsoport ismétlése

A REPT, ENDM direktívapárral egy ismétlendő utasításcsoportot lehet definiálni a következőképpen:

REPT *kifejezés*
utasítások
ENDM

Az ismétlések számát a "*kifejezés*" értéke határozza meg. A kifejezés értéke 16 biten ábrázolható, előjel nélküli szám kell, hogy legyen. Az ismétlési tényező az aktuális alapértelmezés szerinti számrendszerben értendő. Az utasítások tetszőleges, helyesen megadott makró-assembly utasítások lehetnek.

Példa:

```
a   =    0
    REPT5
a   =    a+2
    DW   a
    ENDM
```

A REPT direktíva hatására az = direktíva és a DB direktíva 5-ször ismétlődik. Ennek hatására 5 szónyi adat generálódik a 2,4,6,8,10 értékekkel.

5.3.10 Az IRP és az ENDM direktíva - utasításcsoport ismétlése paraméterhelyettesítéssel

Az IRP, ENDM direktívapárral egy ismétlendő utasításcsoportot definiálhatunk. Az ismétlések számát az IRP direktívában megadott paraméterlista elemeinek száma határozza meg. Az ismétlendő utasításcsoport definiálása a következőképpen történik:

```
IRP helyettesítendő paraméter, <helyettesítő paraméter,,,>
utasítások
ENDM
```

Az utasításcsoport ismétléseinek számát a csúcsos zárójelek között megadott "*helyettesítő paraméter*"-er száma határozza meg. A "*helyettesítendő paraméter*"-rel azt az objektumot jelöljük ki, amelyet az IRP és az ENDM közötti utasításokban való minden előfordulásakor helyettesíteni akarunk az aktuális helyettesítő paraméterrel. A "*helyettesítő paraméter*" tetszőleges, helyesen megadott szimbólum, sztring, numerikus vagy karakter-konstans lehet. Tetszőleges számú helyettesítő paraméter megadható. Több helyettesítő paraméter megadása esetén a paramétereket veszszővel kell elválasztani egymástól. Az "*utasítások*" tetszőleges makró-assembly utasítások lehetnek, amelyekben a helyettesítendő paraméter akárhányszor előfordulhat.

Amikor a fordító egy IRP direktívával találkozik a forrásprogramban, akkor annyiszor másolja be az IRP és az ENDM közötti utasítássorozatot, ahány helyettesítő paramétert megadtunk. Az utasítássorozat első másolatában a helyettesítendő paraméter minden előfordulását az elsőként megadott helyettesítő paraméterrel írja felül, a második másolatban a másodikként megadott helyettesítő paraméterrel, és így tovább. Ha a helyettesítő paraméterek listájában üres paraméter is van, akkor a megfelelő másolatban a helyettesítendő paraméter helye üresen marad. Abban az esetben, ha a helyettesítendő paraméterek listája üres, a fordító nem veszi figyelembe az IRP direktívát és egyetlen utasítást sem másol be.

Az IRP direktíva makró-definíció belsejében is előfordulhat. Ha az itt megadott IRP direktíva helyettesítő paramétere megegyezik a makró formális paraméterével, akkor ezt a formális paramétert kell az IRP direktívában csúcsos zárójelek között megadni. Tekintsük például a következő makró-definíciót:

```
adat  MACRO a
      IRP d,<a>
      DW  d
      ENDM
      ENDM
```

Ha a makrót a következőképpen hívjuk:

```
adat  <1,2,3>
```

akkor a makró kifejtése:

```
IRP d,<1,2,3>
DW  d
ENDM
```

Ebből látható, hogy a makró kifejtésekor a hívásban megadott csúcsos zárójeleket elhagyja a fordító.

Az IRP direktívát természetesen nemcsak makró-definíció belsejében használhatjuk. Az

```
IRP d,<1,2,3>
DW d
ENDM
```

hatására a fordító a következő utasítássorozatot generálja:

```
DW 1
DW 2
DW 3
```

5.3.11 Az IRPC és az ENDM direktíva

Az IRPC, ENDM direktívapárral egy ismétlendő utasításcsoportot definiálhatunk. Az ismétlések számát az IRPC direktívában megadott karaktersorozat karaktereinek száma határozza meg. Az ismétlendő utasításcsoport definiálása a következőképpen történik:

IRPC *helyettesítendő paraméter,karaktersorozat*
utasítások
ENDM

Az utasításcsoport ismétléseinek számát a sztring karaktereinek száma határozza meg. A "*helyettesítő paraméter*"-rel azt az objektumot jelöljük ki, amelyet az IRPC és az ENDM közötti utasításokban való minden előfordulásakor helyettesíteni akarunk a string aktuális karakterével. A "*karaktersorozat*"-ban betűk, számjegyek és egyéb karakterek lehetnek. Ha a karaktersorozatban szóköz, vessző vagy egyéb elválasztó karakter is szerepel, akkor a sztringet csúcsos zárójelek között kell megadni. Az "*utasítások*" tetszőleges makró-assembly utasítások lehetnek, amelyekben a helyettesítendő paraméter akárhányszor előfordulhat.

Amikor a fordító egy IRPC direktívát talál a forrásprogramban, akkor annyiszor másolja be az IRPC és az ENDM között álló utasítássorozatot, amennyi az IRPC direktívában megadott string karaktereinek száma. Az utasítássorozat első másolatában a helyettesítendő paraméter minden előfordulását a string első karakterével írja felül, a második másolatban a string második karakterével, és így tovább.

Példa:

```
IRPC a,123
DW a
ENDM
```

hatására a fordító 3 db DW direktívát generál:

```
DW 1
DW 2
DW 3
```

5.3.12 Az EXITM direktíva

Az EXITM direktívát makrókifejtésből vagy egy utasítássorozat ismétlődő kifejtésekből való kilépésre használjuk.

Alakja:

```
EXITM
```

A kilépés után a fordítás a makróhívás vagy ismétlési sorozat utáni első utasítással folytatódik. Az EXITM direktívát általában a feltételes (IF) direktívákkal együtt arra használjuk, hogy a makrókifejtést vagy az utasítássorozat ismétlését vezéreljük.

Ha a fordító egy EXITM direktívát talál, akkor azonnal abbahagyja a makró kifejtését vagy az ismétlődő utasítássorozatok generálását, és a további utasításokat nem dolgozza fel. Ha az EXITM olyan makróban vagy ismétlési blokkban szerepel, amelyet egy másik makróban vagy ismétlési blokkban definiáltunk, a kifejtés a tartalmazó makróban illetve ismétlési csoportban folytatódik.

A feltételes direktívák ismertetése e fejezet következő pontjában található.

Példa:

```
makróMACRO param
        utasítás
        .
        .
        .
        IF kifejezés
          EXITM
        ELSE
          utasítás
        ENDIF
        ENDM
```

Az itt definiált makró hívásakor a fordító feldolgozza az IF direktíva előtti utasításokat, majd kiértékeli az IF direktívában megadott kifejezést. Ha a kifejezés értéke nulla, akkor az EXITM direktívát dolgozza fel, és abbahagyja a makró kifejtését.

5.3.13 Makró-megjegyzések

A makró-megjegyzések olyan magyarázó szövegek, amelyek a makródefinícióban állnak, de a makró kifejtésekor nem kerülnek be a programba.

Alakja:

;;megjegyzés

A két pontosvessző utáni szöveget a fordító nem veszi figyelembe. Ez a szöveg csak a forrás-listában, a makró definíciójában jelenik meg.

A makró-assembly program egy pontosvesszővel kezdődő megjegyzései, a makró-megjegyzésektől eltérően, bekerülhetnek a makró minden kifejtésébe, ha megadjuk a .XALL direktívát (ld. ebben a fejezetben a 4.6 pontot).

5.3.14 Feltételes fordítás

A makró-assembly nyelvben feltételes direktívákat is lehet használni, amelyekkel feltételes fordítást valósíthatunk meg. A feltételes fordítás azt jelenti, hogy egy utasítássorozatot csak akkor fordítunk le, ha egy adott feltétel teljesül. A feltételes direktívák a következők:

IF
IFE
IF1
IF2
IFDEF
IFNDEF
IFB
IFNB
IFIDN
IFDIF
ELSE
ENDIF

Az IF, ELSE és ENDIF direktívák határolják azokat az utasításokat, amelyeket feltételesen akarunk fordítani. A feltételes blokkokat a következőképpen kell megadni:

IF [*feltétel*]
 utasítások
[**ELSE**
 utasítások]
ENDIF

ahol az utasítások tetszőleges macro-assembly utasítások lehetnek, beleértve újabb IF direktívákat is. Az ELSE direktíva használata nem kötelező.

A macro-assembler csak akkor fordítja le az IF direktíva után lévő utasításokat, ha az adott IF direktívának megfelelő feltétel teljesül. Ha ELSE direktívát is megadtunk, akkor a feltétel teljesülése esetén a blokk utolsó lefordítandó utasítása az az utasítás, amely közvetlenül az ELSE előtt áll. Az ELSE direktívát követő utasítások fordítása csak akkor következik be, ha az IF direktívához tartozó feltétel nem teljesül. A feltételes blokk végét az ENDIF direktíva jelzi. Egy IF direktívához legfeljebb egy ELSE direktíva használata megengedett.

Az IF direktívák legfeljebb 255 szint mélységig ágyazhatók egymásba. Az egyértelműség biztosítása miatt a fordító minden ELSE direktívát a hozzá legközelebbi előző IF direktívához tartozónak tekint.

5.3.14.1 Az IF és az IFE direktíva

Az IF és az IFE direktíva alakja:

IF *kifejezés*
IFE *kifejezés*

Az IF és az IFE direktíva kiértékeli a "*kifejezés*"-t. Az IF direktíva szerint akkor teljesül a feltétel, ha a kifejezés értéke nem nulla. Az IFE direktíva akkor találja igaznak a feltételt, ha a kifejezés értéke nulla.

Az IF és az IFE direktívában olyan kifejezést kell megadni, amelynek konkrét értéke kiszámítható és nem tartalmaz postdefinit hivatkozást.

Példa:

```
IF v-AH
v = v + 1
ENDIF
```

5.3.14.2 Az IF1 és IF2 direktíva

Az IF1 és az IF2 direktíva alakja:

IF1
IF2

Az IF1 és az IF2 direktíva az aktuális fordítási menetet vizsgálja. Az IF1 direktíva hatására a megfelelő utasítássorozat csak az első menetben fordítódik le, az IF2 hatására csak a második menetben. Az IF1 és IF2 direktívának nincs argumentuma.

5.3.14.3 Az IFDEF és az IFNDEF direktíva

Az IFDEF és az IFNDEF direktíva azt vizsgálja, hogy a megadott "azonosító" egy előzőleg már definiált címke, változó vagy szimbólum. Az IFNDEF direktíva szerint akkor teljesül az adott utasítássorozat fordításához szükséges feltétel, ha az "azonosító" még nem definiált.

Tetszőleges "*azonosító*" megadható. Ha az azonosító előre való hivatkozás, akkor az első fordítási menetben definiálatlannak, a másodikban definiáltnak minősül. Ez a fázishibák gyakori oka.

Példa:

```
IFNDEF változó
változó DQ ?
ENDIF
```

5.3.14.4 Az IFB és az IFNB direktíva

Az IFB és az IFNB direktíva alakja:

IFB <*argumentum*>
IFNB <*argumentum*>

Az IFB és az IFNB direktíva az adott "*argumentum*"-ot vizsgálja. Az IFB direktíva hatására akkor történik meg az utasítássorozat fordítása, ha az argumentum üres. Az IFNB direktíva akkor tekinti a feltételt igaznak, ha az argumentum nem üres.

Az "argumentum" tetszőleges azonosító, szám, vagy kifejezés lehet. Kötelező csúcsos zárójelek közé tenni.

Az IFB és IFNB direktívákat főként makró-definíciókban célszerű használni. Az IFB és IFNB segítségével a makróhívásban átadott paraméterek szerint vezérelhetjük a makró utasításainak feltételes kifejtését. Ekkor az "argumentum" a MACRO direktívában megadott egyik formális paraméter.

Példa:

IFB <fp>

Az IFB direktíva a "fp" argumentumot vizsgálja, amely egy makró-definícióban formális paraméter. Ha a makró hívásakor nem adtunk ehhez a formális paraméterhez aktuális paramétert, akkor a megfelelő utasítássorozat lefordítódik.

5.3.14.5 Az IFIDN és az IFDIF direktíva

Az IFIDN és az IFDIF direktíva alakja:

IFIDN <*argumentum1*>,<*argumentum2*>
IFDIF <*argumentum1*>,<*arguméntum2*>

Az IFIDN és az IFDIF direktíva az "*argumentum1*"-et összehasonlítja az "*argumentum2*"-vel. Az IFIDN direktíva hatására akkor fordítódik le a megfelelő utasítássorozat, ha a két argumentum azonos, az IFDIF direktíva hatására akkor, ha a két argumentum **különböző**.

Az argumentumok tetszőleges azonosítók, számok vagy kifejezések lehetnek. A két argumentum akkor azonos, ha karakterről-karakterre megegyeznek. **A csúcsos zárójelek kiírása kötelező.**

Az IFIDN és az IFDIF direktívákat főként makró-definíciókban célszerű használni. Az IFIDN és az IFDIF segítségével a makróhívásban átadott paraméterek segítségével vezérelhetjük a makró utasításainak feltételes kifejtését. Az ilyen esetekben megadott argumentumoknak szerepelniük kell a MACRO direktíva formális paraméter-listájában.

Példa:

IFIDN <a>,<b>

Az IFIDN direktíva összehasonlítja az "a" és "b" argumentumot. Ha ezek egy makródefiníció formális paraméterei, és makróhíváskor "a"-hoz és "b"-hez azonos aktuális paramétert rendeltünk, akkor a megfelelő utasítássorozat lefordítódik.

5.4 A macro-assembler file-kezelésének vezérlése

A makró-assembler lehetőséget nyújt olyan direktívák használatára is, amelyek a fordítás során a makró-assembler által olvasandó és előállítandó forrás-, object- és lista-file-ok kezelését vezérlik. A következő direktívák adhatók meg a file-kezelés vezérlésére:

INCLUDE	Forrás-file bemásolása
.RADIX	A forrás-file-ban lévő számok alapértelmezés szerinti számrendszerének beállítása
%OUT	Üzenetek kiírása a konzolra
NAME	Modulnév beírása az object-file-ba
TITLE	A programlista címének megadása
SUBTTL	A programlista alcímének megadása
PAGE	A programlista lapméretének beállítása, feltétel nélküli lapdobás előírása
.LIST	Utasítások kiírása a programlistába
.XLIST	Utasítások elhagyása a programlistából
.LFCOND	Utasítások listázása, ha a feltétel nem teljesül az IF direktívákban
.SFCOND	Utasítások elhagyása a programlistából, ha a feltétel nem teljesül az IF direktívákban
TFCOND	Feltételes blokkok listázási módja alapértelmezésének beállítása
.LALL	Makrókifejtések listázása
.SALL	Makrókifejtések elhagyása a programlistából
.XALL	Megjegyzések kihagyása a makrók listájából
.CREF	Szimbólumok beírása a keresztreferencia-file-ba
.XCREF	Szimbólumok kihagyása a keresztreferencia-file-ból

5.4.1 Az INCLUDE direktíva

Az INCLUDE direktíva segítségével egy adott forrás-file-t másolhatunk be fordítás közben az aktuális forrás-file-ba. Az INCLUDE direktíva alakja:

INCLUDE *file-név*

A "file-név" egy létező file neve. Ha a file nem abban a directory-ban található, amely a fordításkor aktuális, akkor meg kell adni a file-t tartalmazó directory-t. Ha a makró-assembler nem találja az adott file-t, akkor hibajelzést ad és a fordítás megszakad.

Amikor a fordító egy INCLUDE direktívát talál, megnyitja az adott file-t és azonnal elkezdi fordítani a megnyitott file forrás-utasításait. Miután a file minden utasításának a fordítása megtörtént, a makró-assembler az INCLUDE direktívát követő utasítás feldolgozásával folytatja tevékenységét.

Az INCLUDE direktívák egymásba ágyazhatók, ami azt jelenti, hogy a direktívában megadott file is tartalmazhat INCLUDE direktívákat. A programlistában a macro-assembler "c" betűvel jelzi az INCLUDE direktíva hatására beillesztett utasításokat.

Példák:

```
INCLUDE struktúrák
INCLUDE \dir\struktúrák
INCLUDE C:\dir\struktúrák
```

5.4.2 A TITLE direktíva

A TITLE direktívát a programlista címének meghatározására használhatjuk a következőképpen:

TITLE *szöveg*

A direktíva hatására a makró-assembler kiírja a "szöveg"-et a forráslista minden új lapjának első sorába. A "szöveg" tetszőleges karakterekből állhat és legfeljebb 60 karakter hosszúságú lehet. Egy modulban legfeljebb egy TITLE direktíva adható meg.

Ha a modul nem tartalmaz NAME direktívát, akkor a fordító a TITLE direktívában megadott "szöveg" első 6, szóköztől különböző karakterét tekinti modulnévnek.

Példa:

```
TITLE 1. program
```

5.4.3 A SUBTTL direktíva

A SUBTTL direktíva segítségével a programlista alcímét határozhatjuk meg. Alakja:

SUBTTL *szöveg*

A SUBTTL direktíva hatására a makró-assembler a TITLE direktívával a programlistába írt *szöveget* követő sorba kiírja a SUBTTL direktívával megadott alcímet. Az alcím a programlista minden új lapján megjelenik. A szöveg tetszőleges karakterekből állhat. A fordító csak az alcím első 60 karakterét veszi figyelembe. Ha a SUBTTL direktívával nem adunk meg szöveget, akkor az alcím sora üresen marad. Egy programban tetszőleges számú SUBTTL direktíva megadható. Minden újabb SUBTTL direktíva új szöveggel helyettesíti az előző alcímet.

Példák:

```
SUBTTL ellenőrzések
SUBTTL
```

5.4.4 A PAGE direktíva

A PAGE direktívával beállíthatjuk a programlista lapméretét, növelhetjük a szakasz-számot vagy lapozhatunk a programlista készítése közben. Ezen tevékenységek végrehajtásához a PAGE direktíva a következő módokon adható meg:

PAGE [*sorszám*],*sorszélesség*
PAGE +
PAGE

A *"sorszám"* az egy lapon kiírható sorok maximális számát, a *"sorszélesség"* az egy sorban kiírható karakterek maximális számát határozza meg. A sorok száma a [10,255] intervallumba kell, hogy essen. Alapértelmezés szerint egy lapra 50 sor kerül. A sorszélességnek a [60,132] intervallumba kell esnie. Alapértelmezés szerint egy sorba 80 karakter kerül. A sorszélességet úgy is megadhatjuk, hogy a sorok számát nem írjuk ki, vagyis az aktuális beállított sorszámot kérjük a programlista készítéséhez.

A vessző megadása azonban ebben az esetben is kötelező. Ha a PAGE direktíva másodikként felírt alakját használjuk, akkor a szakasz száma 1-gyel nő és a szakaszon belüli lapszám 1-re állítódik be. A program listázásakor a lapok számozásának alakja:

szakasz száma - szakaszon belüli lapszám

Alapértelmezés szerint a lapok számozása 1-1-gyel kezdődik. A PAGE direktíva argumentum nélküli formájával lapemelést hajthatunk végre programlista készítése közben. A lista-file-ba lapemelés karakter kerül, valamint cím-et és alcím-sort generál.

Példák:

```
PAGE 40,121
```

Hatására a programlista egy lapjára 40 sor, egy sorába 121 karakter kerül.

```
PAGE ,132
```

Hatására a programlista sorszélessége 132 karakter lesz, az egy lapon kiírt sorok száma változatlan marad.

```
PAGE +
```

A direktíva növeli a szakasz számát és a szakaszon belüli lapszámot 1-re állítja.

```
PAGE
```

Lapemelést hajt végre.

5.4.5 A .LIST és a .XLIST direktíva

A .LIST és a .XLIST direktívával azt határozhatjuk meg, hogy mely forrás-sorok jelenjenek meg a programlistában. A .LIST illetve a .XLIST direktíva alakja:

```
.LIST
.XLIST
```

A .XLIST direktíva megszünteti a további forrás-soroknak a listába való helyezését. A .LIST direktíva hatására a direktíva utáni utasítások bekerülnek a listába. Ezeket a direktívákat általában együttesen használjuk abból a célból, hogy egy adott forrás-file valamely része ne jelenjen meg a program listájában.

A .XLIST direktíva minden egyéb, listázással kapcsolatos direktívát felülbírál.

5.4.6 A .LALL, a .XALL és a .SALL direktíva

A .LALL, a .XALL és a .SALL direktívával a forrás-file-ban kifejtett makrók utasításainak listázását vezérelhetjük. A direktívák alakja:

```
.LALL
.XALL
.SALL
```

A makró-assembler mindig listázza a teljes makró-definíciót, de a makró-kifejtés listázását a programozó határozhatja meg. A .LALL direktíva hatására a makró-assembler a makró-kifejtés minden utasítását listázza a makró-megjegyzések (;;szöveg) kivételével. A normál assembly megjegyzések (;szöveg) megjelennek a listában. A .XALL direktíva hatására a makró-kifejtésből csak azok az utasítások kerülnek a listába, amelyek adatot vagy kódot generálnak. Így a makró-assembler a megjegyzéseket és pl. a változó = változó + 1 utasításokat nem listázza. A .SALL direktíva elnyomja a makró-kifejtés összes utasításának listázását.

Ha egyik direktívát sem adjuk meg, akkor .XALL az alapértelmezés.

5.4.7 A .SFCOND, a .LFCOND és a .TFCOND direktíva

A .SFCOND, a .LFCOND és a .TFCOND direktívával feltételes blokkok listázását vezérelhetjük. A direktívák alakja:

```
.SFCOND
.LFCOND
.TFCOND
```

A .SFCOND direktíva hatására a fordító nem listázza a továbbiakban azokat a feltételes blokkokat, amelyekben az IF feltétel nem teljesül. A .LFCOND direktíva visszaállítja ezeknek a blokkoknak a listázását.

A .TFCOND direktívával a feltételes blokkok listázási módjának az alapértelmezését állíthatjuk be. A .TFCOND direktíva hatása attól függ, hogy a fordító hívásakor megadtuk-e a /X opciót (ld. VII. fejezet 2. pont). Ha igen, akkor a .TFCOND direktíva elnyomja az olyan feltételes blokkok listázását, amelyekben a feltétel nem teljesül. Ha a fordító hívásakor nem adtuk meg a /X opciót, akkor a .TFCOND direktíva hatására a "hamis" feltételes blokkok listázása lesz az alapértelmezés.

Példák:

```
.SFCOND
IF 0
        ; Nem listázza az utasításokat
ENDIF
.LFCOND
IF 0
        ; Listázza az utasításokat
ENDIF
```

5.4.8 A .RADIX direktíva

A .RADIX direktíva segítségével beállíthatjuk a forrás-file-ban lévő számok alapértelmezés szerinti számrendszerét. A .RADIX direktíva alakja:

.RADIX *alapszám*

Az "*alapszám*" azt határozza meg, hogy a forrás-file-ban használt számok milyen számrendszerben értelmezendők. A számrendszer alapja 2-16 lehet. Általában a következő értékeket szokás használni:

2	bináris
8	oktális
10	decimális
16	hexadecimális

A .RADIX direktívában megadott "*alapszám*"-ot mindig decimális számként értelmezi a fordító, attól függetlenül, hogy mi az aktuális alapértelmezés szerinti számrendszer.

Példák:

```
.RADIX 8
.RADIX 10
```

A **.RADIX direktíva hatása a DD, DQ és DT direktívákra nem terjed ki.** A DD, DQ és DT direktívákban megadott számokat a fordító mindig decimális számoknak tekinti. A .RADIX direktíva hatása a B és D opcionális típusjelek megadása mellett sem érvényesül. Ha valamely egész szám után a B vagy D típusjel áll, akkor a számot a B (bináris) vagy D (decimális) számrendszer által megengedett jegyekkel kell felírni, abban az esetben is, ha az aktuális alapértelmezés szerinti számrendszer hexadecimális. Például

```
5ACB
```

numerikus érték hibás, mert 5, A és C nem bináris számjegyek.

5.4.9 A .CREF és a .XCREF direktíva

Az .CREF és a .XCREF direktívával a fordító keresztreferencia-file-jának előállítását befolyásolhatjuk. A direktívák alakja:

.CREF
.XCREF [*azonosító*[,,,]]

A .XCREF direktíva hatására a fordító nem készít keresztreferenciát a címkékről, változókról és egyéb szimbólumokról. A .CREF direktíva visszaállítja a fordító keresztreferencia-generáló tevékenységét.

Ha a .XCREF direktívában azonosítókat is felsorolunk, akkor az itt megadott *azonosítók* **kivételével** minden címke, változó és egyéb szimbólum bekerül a keresztreferencia-file-ba.

5.4.10 A NAME direktíva

A NAME direktívával az aktuális modul nevét határozhatjuk meg. Alakja:

NAME *modulnév*

A *modulnév* betűk és számjegyek tetszőleges kombinációja. Bármilyen hosszúságban megadható, de csak az első 6 karakter szignifikáns. A modulnévnek egyedinek kell lennie és nem lehet foglalt név.

A modulnevet a futtatható programot előállító szerkesztő (lásd a 8. fejezetet) használja, amikor hibajelzéseket ír ki.

Ha a NAME direktívát nem használjuk, akkor a fordító a TITLE direktívával megadott szöveg első 6 karakterét tekinti modulnévnek. **Abban az esetben, ha TITLE direktívát sem adtunk meg, az alapértelmezés szerinti modulnév "A".**

5.4.11 A %OUT direktíva

A %OUT direktíva hatására a fordító egy üzenetet jelenít meg a felhasználó terminálján. A direktíva alakja:

%OUT *szöveg*

A direktíva használata főleg hosszú programok fordításakor célszerű, mivel így információt kaphatunk a fordítási folyamat előrehaladásáról.

A %OUT direktíva outputot állít elő mindkét fordítási menetben. Az IF1 és IF2 direktívát annak eldöntésére is használhatjuk, hogy kiírjuk-e a képernyőre az üzenetet.

Példa:

```
IF2
  %OUT  Már a második menetnél tart
ENDIF
```

5.5 Magasszintű nyelvből hívott assembly rutinok

A legtöbb magasszintű nyelv fordítója lehetőséget biztosít a felhasználó által írt assembly nyelvű rutinok beépítésére. Az assembly rutin alkalmazásának szabályai **nem egységesek**, de az adott fordító felhasználói kézikönyvében mindig találunk ismertetést az assembly rutinok használatának feltételeiről. Ilyen feladatra csak gyakorlott assembly-programozók vállalkozzanak!

Mindig pontosan nézzünk utána a belső adatábrázolási formátumoknak és a paraméterátadási mechanizmus működésének. A paraméterátadás általában a vermen keresztül történik (de ez nem törvény), a paramétereknek vagy az értéke, vagy a címe férhető hozzá az assembly rutinból. A regisztereket belépéskor mentsük el, és visszatérés előtt állítsuk vissza (kivéve, ha a paraméterátadás regiszterben történik). Különösen fontos tisztázni, hogy a vermet milyen állapotban kell visszaadni (az ott kapott paramétereket üríteni kell vagy nem, a visszaadandó paramétereket hogy kell elhelyezni... stb)!

Az elkészített assembly rutint teszteljük le még assembly környezetben, csak ezután próbáljuk meg a magasszintű nyelvbe beilleszteni!

6. A MASM fordító használata

6.1 Bevezetés

A Microsoft Macro Assembler (MASM) az assembly nyelvű forrásprogramokat áthelyezhető tárgyprogrammá (.OBJ) alakítja, amelyekből a LINK szerkesztőprogrammal készíthetünk végrehajtható (.EXE) file-t, amely az operációs rendszerből közvetlenül elindítható.

6.2 Forrás-file fordítása

A szövegszerkesztővel megírt forrásfile-okat a MASM segítségével fordíthatjuk le. A fordítót a következő paranccsal indíthatjuk el:

MASM [[*kapcsolók*]*forrás*,[*tárgy*],[*lista*],[*crf*]]

kapcsolók	a fordító működését befolyásoló paraméterek, leírásuk a 3. pontban található
forrás	a forrásszöveget tartalmazó file neve
tárgy	a tárgykód-file neve (ez lesz a fordítás eredménye)
lista	a lista-file neve
crf	a keresztreferencia file neve

A parancssorban akár minden paraméter elhagyható, ekkor a MASM kérdéseket tesz fel minden megadható paraméterre vonatkozóan.

Amikor paraméterrel indítjuk el az assemblert, a forrás megadása kötelező és a többi filenév elhagyható (de a vesszők nem!), az elhagyott nevek helyett a rendszer a forrás nevét helyettesíti be, de csak a név helyett mert a kiterjesztésekre konvenciókat alkalmaz (ha nem adunk meg kiterjesztést, mindig helyettesítődik az alapfeltételezés). A konvenciók és alapértelmezések a következők:

forrás	a kiterjesztésre .ASM
tárgy	file-név azonos a < forrás > -sal, kiterjesztése .OBJ
lista	file-név NUL (nem készül lista), kiterjesztése .LST
crf	file-név NUL (nem készül keresztreferencia) kiterjesztése .CRF

Ha a parancssorban bármelyik vessző helyén pontosvesszőt írunk, a MASM a további paraméterek helyén az alapfeltételezést helyettesíti.

Példák:

```
MASM forr;
```

Az assembler program a FORR.ASM file-ban van, FORR.OBJ lesz az object file neve, lista és kereszthivatkozási lista nem készül.

```
MASM forr,,;
```

Az assembly program a FORR.ASM file-ban van, FORR.OBJ lesz az object és FORR.LST a lista file neve, kereszthivatkozási lista nem készül.

```
MASM forr.src,forr.trg,lista.lis,xr.xrf
```

Az assembler program a FORR.SRC file-ban van, az object a FORR.TRG file-ba, a lista a LISTA.LIS nevűbe és a keresztihivatkozás az XR.XRF file-ba kerül.

Ha nem adtunk meg minden paramétert, de nem pontosvesszővel zártuk le a sort, a hiányzó paraméterekre rákérdez a MASM ugyanúgy, mint amikor egyáltalán nem adunk paramétert a fordító indításánál. A kérdések a következők:

```
Source filename [.ASM]:forr
Object filename [forr.OBJ]:<ENTER>
Source listing  [NUL.LST]:<ENTER>
Cross reference [NUL.CRF]:<ENTER>
```

Az alapértelmezések behelyettesítése most is megtörténik, a fenti példában az assembly program a FORR.ASM file-ban van, az object a FORR.OBJ-be kerül, lista és keresztihivatkozás nem készül.

Bármelyik filenév kiegészíthető meghajtó azonosítóval és elérési úttal is. Kapcsolók bármelyik sor végén megadhatók.

Hasonlóan a parancssorhoz, ha az .ASM file megadása után bárhol pontosvesszőt írunk, a további kérdéseket nem teszi fel a MASM és az alapértelmezéseket behelyettesíti.

6.3 A MASM kapcsolói

6.3.1 /D, az első menet listája

A kapcsoló hatására a fordítás mindkét menetében listát készít a MASM, amiben a hibajelzések is szerepelnek. Erre akkor van szükség, ha a fázishiba (Phase Error) okát nem találjuk. Első menetbeli lista csak akkor készül, ha a lista-file-t megadjuk, ha nem, csak az első menet hibajelzései íródnak ki (azok is amelyeket a második menet felold!).

6.3.2 /ML, kis- és nagybetűk megkülönböztetése a lokális szimbólumokban

Ha ezt a kapcsolót nem adjuk meg, a MASM minden szimbólumot nagybetűsre alakít. Ha ez a tulajdonság nem kívánatos, ezzel a kapcsolóval kikapcsolhatjuk a konverziót. Ebben az esetben a 'Cimke' és a 'cimke' azonosító különböző lesz!

6.3.3 /MX, kis- ésnagybetűk megkülönböztetése a globális szimbólumokban

Ha ezt a kapcsolót nem adjuk meg, a MASM minden globális szimbólumot nagybetűssé alakít. Ha ez a tulajdonság nemkívánatos, ezzel a kapcsolóval kikapcsolhatjuk a konverziót. Erre akkor van szükség, ha olyan tárgymodullal akarjuk összeszerkeszteni programunkat, amely megkülönbözteti a kis- és nagybetűket.

6.3.4 /X, a feltételes fordítás hamis ágának listázása

Ha ezt a kapcsolót nem adjuk meg, a MASM azokat a forrássorokat nem listázza, amelyek egy feltétel hamis ágában vannak. Ez az IF utasítás minden formájára igaz. Amikor a kapcsolót megadjuk, a hamis ágak forrássorai bekerülnek a listába.

Az X kapcsoló hatását befolyásolják a forrásszövegben elhelyezett direktívák is, ezek:

.SFCOND .LFCOND .TFCOND

A .SFCOND és a .LFCOND az X kapcsolótól függetlenül vezérlik a listázást. A .TFCOND minden előfordulása ellenkezőjére változtatja az eddigi hatását az X kapcsolónak (első .TFCOND után a hamis ágakat nem listázza, a második után újra listázza... stb.).

direktíva	/X hatása
.SFCOND	hatástalan, nem listáz
.LFCOND	hatástalan, listáz
.TFCOND	nem listáz
nincs	listáz

Mindez természetesen csak akkor igaz, ha kértünk listát a fordítótól (megadtuk a lista-file nevét).

6.3.5 /R, kódképzés az aritmetikai processzor számára

Ha a direktívát megadjuk, a fordító a 8087/80287 aritmetikai processzor mnemonikjait utasításokká fordítja. Az így fordított program csak olyan gépen fut, amelyben van aritmetikai processzor!

6.3.6 /E, kódképzés a lebegőpontos emulátor számára

Ha a kapcsolót megadjuk, a 8087/80287 aritmetikai processzor mnemonikjai helyett olyan hívásokat generál a fordító, amelyek 8086-ot utasításokkal emulálják az aritmetikai processzor műveleteit. Ezeket a rutinokat egy különálló emulátorkönyvtár tartalmazza, ezt hozzá kell szerkeszteni a programhoz, mert ha nem tesszük, nem definiált szimbólumok keletkeznek a programban. Emulátor könyvtárat a magasszintű programnyelvek programcsomagjai általában tartalmaznak.

6.3.7 /A, szegmensek abc sorrendben

Amikor a kapcsoló nincs megadva, a szegmensek a tárgykódbeli előfordulásuk sorrendjében kerülnek lefordításra, a kapcsoló megadásával a szegmensnevek szerinti abc sorrendben.

6.4 A lista-file

Ha a lista-file nevét megadjuk a MASM indításakor, a forrásprogramról részletes listát készít a MASM. A listában szerepel minden utasítás címe, tárgykódja és maga az utasítás is. A forráslista végén kigyűjtött listát kapunk a használt makrók, struktúrák, rekordok, szegmensek, csoportok és egyéb szimólumok azonosítóiról és jellemzőiről. Az azonosítók csoportonként névsorba vannak rendezve.

6.4.1 A forrássorok

A forrássorok alakja a listában:

[*sorszám*] *offset kód utasítás*

sorszám	a forrássorok 1-től kezdődő folyamatos sorszáma, csak akkor szerepel ha .CRF file-t megadtunk
offset	az utasítás szegmensen belüli címe
kód	az utasításnak megfelelő object kód vagy adat. Amennyiben ez nem végleges értékű, megadja a kiszámítás módját.
utasítás	egyszerű esetben a forrásszöveg egy sora, makró-kifejtés esetén ez eltérhet a forrásszövegtől

A megtalált hibákat a hibás sor után írt hibajelzéssel jelzi a MASM.

A "kód" mezőben használt speciális jelölések a következők:

Jel	**Jelentése**
R	relokálandó cím, végleges értékét a szerkesztő számítja ki
E	külső (EXTRN) cím, értékét a szerkesztő adja meg
-	szegmens (csoport) cím, értékét a szerkesztő számítja ki
=	EQU vagy = direktíva
nn:	az utasítás szegmensmódosítást ír elő
nn/	az utasításban REP vagy LOCK prefix van
nn[xx]	DUP kifejezés
+	a sor makrókifejtésel keletkezett
C	a sor egy INCLUDE file sora

6.4.2 A makrótáblázat

Abc sorrendben felsorolja a definiált makrókat és a makródefiníció forrássorainak számát.

6.4.3 A struktúra- és rekordtáblázat

A táblázat tartalmazza a programban előforduló struktúrák, rekordok és rekordmezők azonosítóit abc sorrendben.

A struktúra azonosító sorában a Width oszlop a struktúra mérete byte-okban, a # fields Width oszlopban a mezők száma található. A struktúra mezőazonosítók sorában a shift oszlop a mező távolságát adja byte-okban a struktúra kezdetétől.

Rekord esetén a rekordazonosító sorában a Width oszlopban a rekord mérete szerepel bitekben, a # fields Width oszlop a mezők számát adja.

A mezőazonosítók 'Shift' oszlopa a mező első bitjének távolságát adja a rekord legkisebb helyiértékű bitjétől számítva. A # fields Width oszlop a mező szélessége bitekben. A Mask oszlop a mező maszkjának értéke. Az Initial oszlop a mező inicializálási értéke.

6.4.4 A szegmens- és csoporttáblázat

A táblázat oszlopai:

Name	a szegmens, csoport neve
Size	mérete byte-okban
Align	az igazítás típusa
Combine	a kombináció típusa
Class	a kombinációs osztály neve

Az igazítás típusa BYTE, WORD, PARA és PAGE lehet. Ha nem adtuk meg, PARA lesz. A kombináció típusa PUBLIC, STACK, MEMORY és NONE lehet. Ha nem adtuk meg, NONE lesz. A kombinációs osztály neve az a név, amit a szegmens definíciójában megadtunk.

6.4.5 A szimbólumtáblázat

A programban használt szimbólumok abc sorrendbe rendezett listája, új kezdőbetűnként egy üres sorral elválasztva. A táblázat oszlopai:

Name	név
Type	típus
Value	érték
Attr	tulajdonságok

A típus oszlopban a következők szerepelhetnek:

L NEAR	közeli címke
L FAR	távoli címke
L BYTE	byte adat cimkéje
L WORD	szó adat cimkéje
L DWORD	dupla szó adat cimkéje
L QWORD	4 szó adat cimkéje
L TBYTE	10 byte adat cimkéje
N PROC	közeli eljárás
F PROC	távoli eljárás
Number	szám
Alias	egy szimbólum másik változata
Opcode	utasításkód
Text	karakter (sor) konstans

Az érték oszlopban a szimbólum értéke szerepel.

A tulajdonság oszlopban a definiáló szegmens neve, a szimbólum hatásköre (External, Global) vagy eljárás esetén a mérete byte-okban.

6.4.6 Az első menet listája

Ha a /D kapcsolóval előírtuk, a MASM a fordítás első menetében is listát készít. Az első menet listája tele lesz hibajelzésekkel, mert pl. minden előrehivatkozás cimkéje az első menetben a fordítás pillanatában ismeretlen címke, és ezt hibaként jelzi is a fordító. Ez a hiba persze a második menetben megszűnik, hisz akkor már ismert lesz a címke. Makacs fázishibák esetén sokat segíthet ez a tulajdonsága a fordítónak, mert a két fázisban eltérő lesz a kapott hibajelzés.

Pl: egy mintaprogram:

```
code  segment
      jmp cimke
cimke label fare
      cmp al,1
code ends
```

Ezt lefordítva két hibajelzést kapunk:

```
    jmp cimke
Error...57:Illegal size for item

    cimke label far
Error...6: Phase error between passes
```

Most első menetben is listát kérve csak egy hiba lesz:

```
    jmp cimke
Error...9:Symbol not defined
```

Az első menetben a cimke definiálatlan, mert előrehivatkozás, de a cimke típusa nincs megadva, ezért a near alapértelmezést rendeli hozzá a fordító és két byte helyet foglal számára az ugró utasításban. A második menetben már ismert a cimke és tudjuk, hogy far típusú, de az ilyen cimkére ugrás 4 byte-ot igényel és két byte helye van, ez fázishibát okoz. A hiba egy lehetséges feloldása:

```
jmp far ptr cimke
```

7. Programkönyvtár-kezelés a LIB segédprogrammal

7.1 A programkönyvtár és kezelése

A programkönyvtár háttértáron - általában lemezen - tartott programgyűjtemény. Egy gyűjtemény programjai különféle, az adott gépen rendelkezésre álló magasszintű nyelven és assemblerben készültek, és vagy eredeti forrásszöveg alakban, vagy már lefordított, tárgykódú változatban (utóbbiakat nevezik tárgymoduloknak vagy object moduloknak) szerepelnek a gyűjteményben. Ez a fejezet a tárgymodul könyvtárakkal foglalkozik.

A programkönyvtár fenntartásának olyan operációs rendszerek használatakor van elsősorban értelme, amelyek tartalmaznak rendszerprogramokat

- a könyvtárbeli programoknak a felhasználói programokkal való összeépítéséhez, és
- a programkönyvtár karbantartásához.

A PC DOS programszerkesztője, a LINK teljesíti az első követelményt.

Az operációs rendszerhez rendelkezésre áll egy programkönyvtárkezelő, a **Microsoft Library Manager**, a LIB. A LIB tárgymodulokat kezel, és biztosítja a szokásos lehetőségeket:

- a programkönyvtár létrehozása
- a könyvtár programjairól katalógustáblázat létrehozása és karbantartása a könyvtár változásai szerint
- e katalógus kinyomtatása a közösnek deklarált szimbólumokkal együtt
- a könyvtár karbantartása, azaz az egyes programok, modulok *felvétele* a könyvtárba (add), *törlése* (del), *kicserélése* (replace), a felhasználói programokhoz való továbbítása (copy), kérésre a könyvtárból való törléssel együtt (move).

A karbantartó funkciók, akárcsak a PC DOS-hoz mellékelt más rendszerprogramok esetében is, többféle módon elérhetők:

- a LIB hívása után a LIB promptjain
- parancs-soron
- @ file-ba eltett parancsgyűjteményen
- a fenti lehetőségek kombinációján keresztül.

A fejezetben ismertetjük a LIB-hez e három módon való hozzáférés szabályait, és a könyvtár, valamint a tárgymodulok előbb felsorolt kezelési lehetőségeit.

7.2 A LIB funkciói

A LIB funkcióihoz háromféle módon férhetünk hozzá:

- a LIB parancs hívásakor átadott paraméterekkel;
- párbeszédes üzemmódban a LIB kérdéseire adott válaszokkal;
- file-ban előre összeállított parancssorozattal.

A LIB működés közben bármikor megállítható a < Ctrl + C > billentyűk lenyomásával.

A funkciók a könyvtár módosítására szolgálnak. A könyvtár könyvtármodulokból áll, amelyek kiterjesztése alapértelmezés szerint .OBJ. A könyvtárat csak tárgymodulokkal lehet bővíteni. A könyvtár változtatására szolgáló funkciók hívása: a funkció jelét, amely egy adott modulra vonatkozik, közvetlenül (szóköz nélkül) a kérdéses modul neve elé írjuk. A "műveleti" jelek és a hozzájuk tartozó funkciók:

Jel	A funkció feladata
+	Egy modul hozzáadása (ADD)
-	Egy modul törlése (DELETE)
-+	Egy modul helyettesítése azonos nevű modullal (REPLACE)
*	Egy modul átmásolása az aktuális tartalomjegyzékbe (EXTRACT)
-*	Egy modul átírása az aktuális tartalomjegyzékbe, és törlése a könyvtárból (EXTRACT and DELETE)
+	Egy könyvtár hozzáadása (a kiterjesztés .LIB) (JOIN)

Az egyes parancsokra rendre bemutatunk egy-egy példát:

A kiindulási állapot:

Az aktuális lemezegység a C. Ezen van a LIB segédprogram és egy MKT.LIB nevű könyvtár, valamint a következő modulok:

X.OBJ, A.OBJ, B.OBJ, C.OBJ, és T.OBJ.

A C lemezegységen van egy MUNKA tartalomjegyzék, benne egy MOD.LIB nevű könyvtár, melynek elemei a következő modulok:

R.OBJ, S.OBJ, T.OBJ, U.OBJ.

A T.OBJ nevű modulok ugyanannak a rutinnak a változatai. A MUNKA-beli T.OBJ az újabb. A parancsok a táblázatban felírt sorrend szerint:

```
LIB c:\MUNKA\MOD +X
```

Hatása: a MUNKA-tartalomjegyzékbeli MOD könyvtárunk kiegészül az "X" modullal azaz tartalma R,S,T,U,X lesz.

```
LIB C:\MUNKA\MOD -U
```

Hatása : a MOD.LIB tartalma R,S,T,X lesz, az U modul törlődik a könyvtárból.

```
LIB \MUNKA\MOD -+T
```

Hatása: A MOD.LIB-ben levő T.OBJ rutint újra cseréli.

```
LIB \MUNKA\MOD *R
```

Hatása: a MOD.LIB-ből az R.OBJ modult kiírja az aktuális tartalomjegyzékbe.

```
LIB \MUNKA\MOD -*S;
```

Hatása: az S.OBJ modult törli és kiírja az aktuális tartalomjegyzékbe. Így a MOD.LIB tartalma R,T,X lesz.

```
LIB \MUNKA\MOD + X-U- + T*R-*S + MKT.LIB;
```

A LIB ilyen esetben először elvégzi a file-kiírásokat (a * művelet), majd az összes törlést, azután veszi sorra a többi műveletet.

7.2.1 A LIB hívása parancs-sorral

A LIB indítása PC DOS parancs-sorral a következő formában történik:

LIB *könyvtár-file neve*[,*kit*][/**PAGESIZE**:*n*] *parancs*[,*lista-file*][,*output-file*]

A könyvtár-file az a programgyűjtemény, amelyet a LIB-bel kezelünk. Ha kiterjesztését nem adjuk meg, akkor az alapértelmezés .LIB.

A /**PAGESIZE** opció a könyvtár lapméretét határozza meg, amely alapértelmezés szerint az újabb verziókban 16 byte. (A lapméretre állítás lehetőségére még visszatérünk.)

Az egy vagy több "*parancs*"-csal azokat a műveleteket - LIB funkciókat - adjuk meg, amelyeket a könyvtár file-on végre akarunk hajtatni.

A "*lista-file*" a keresztreferencia-lista file neve. Ilyen file csak akkor készül, ha ezt a paramétert megadjuk.

Az "*output-file*" megadására akkor van szükség, ha más néven akarjuk megőriztetni a könyvtár-file új, a LIB végrehajtása utáni formáját. Ha ez a paraméter elmarad, a módosult könyvtár-file neve ugyanaz lesz, mint az eredetié. Az új könyvtár előállítása előtt a LIB az eredeti könyvtárat átnevezi, a könyvtár régi változata új nevének kiterjesztése .BAK lesz. A file-azonosítók írásában a PC DOS konvenciók érvényesek.

Ügyeljünk a parancsok utáni vesszőkre, és ha csak output file-t adunk meg, akkor ","-vel jelezzük, hogy nincs lista-file. A parancssor végét bármely elem után ;-vel jelezhetjük, ekkor a további paraméterekre az alapértelmezések érvényesek.

Példa:

```
LIB rend - + olv;
```

A parancs hatására a LIB a REND.LIB nevű könyvtárban az "olv" modult az aktuális tartalomjegyzékben található OLV.OBJ modullal helyettesíti. A ; jelzi, hogy a lista- és az output-file-ra az alapértelmezés érvényes, vagyis keresztreferencia-lista nem készül, és a módosítás után az új könyvtár a régi REND.LIB néven kerül tárolásra.

```
LIB rend - + olv,rend.LST,rend1.LIB
```

Ezzel a paranccsal a REND.LIB módosítása után létrehozzuk a REND1.LIB könyvtárat, amelyben már az "olv" modul új változata van. Az új könyvtárról REND.LST néven készül keresztreferencialista.

7.2.2 A LIB használata párbeszédes üzemmódban

A LIB működéséhez szükséges információkat a LIB kérdéseire - úgynevezett promptjaira - adott válaszokkal is beírhatjuk.

A LIB párbeszédes módban való használatának lépései:

1. Begépeljük a könyvtárkezelő segédprogram nevét:

LIB

és megnyomjuk az < ENTER > gombot. Ennek hatására a LIB megkérdezi a kezelni kívánt könyvtár nevét (a képernyőn a következő üzenet jelenik meg):

Library name:

2. Az előbbi üzenetre válaszul beírjuk a könyvtár nevét. Ha nem adunk meg kiterjesztést, akkor .LIB az alapfeltételezés. Új könyvtárat nem létező file-név megadásával hozhatunk létre. A LIB visszakérdez, hogy létrehozza-e:

Library file does not exist. Create?

A válasz YES, ha létre akarunk hozni egy új könyvtárat, NO, ha vissza akarunk térni az operációs rendszerhez. Ezután megkérdezi, hogy milyen műveleteket akarunk elvégezni:

Operations?

3. Egy vagy több parancsot írunk be és megnyomjuk az < ENTER >-t. Ha a paranacsok nem férnek el egy sorban, akkor & jel és < ENTER > után a következő sorban folytathatók. Ha csak a könyvtár konzisztenciáját (ennek jelentésére később visszatérünk) akarjuk ellenőrizni, akkor nem adunk meg parancsot.

Az < ENTER > gomb megnyomása után a LIB megkérdezi a keresztreferencia-lista-file nevét:

List file:

4. Ha kérünk keresztreferencia-listát, akkor válaszul a teljes file-nevet írjuk be, mert a kiterjesztésre nincs alapértelmezés. Ha nincs szükség lista-file-ra, elegendő csak az < ENTER > megnyomása. A képernyőn megjelenik az output-file-ra vonatkozó kérdés:

Output library:

5. Ha a módosítás után új néven akarjuk elmenteni a könyvtárat, akkor beírunk egy könyvtárnevet. A helyzet ugyanaz, mint a parancssor esetén: ha adunk output-nevet, a .LIB kiterjesztés itt is alapfeltételezés.
Ha nem adunk nevet, a program akkor is létrehozza az új könyvtárat az eredeti könyvtárnévvel azonos néven, de ezt megelőzően az eredeti könyvtárat átnevezi úgy, hogy nevének kiterjesztése .BAK lesz.

Megjegyzés:

A 2. lépéstől bárhol írhatunk válaszként ;-t. Ekkor a LIB több kérdést nem tesz föl, és az alapértelmezéseket alkalmazza.

A file-azonosítók írására a PC DOS konvenciók érvényesek.

Példa:

```
LIB
Library file:    SZAMOL <ENTER>
Operations:      +OSSZEAD+SZOROZ&<ENTER>
Operations:      +KIVON+OSZT<ENTER>
List file:       <ENTER>
Output Library:  SZAMOL1<ENTER>
```

A korábbi SZAMOL.LIB könyvtárból egy SZAMOL1.LIB nevű könyvtárat készít a beírt 4 modul hozzáadásával, amelyek mindegyike .OBJ kiterjesztésű. E modulok itt is az aktuális tartalomjegyzékből származnak.

```
LIB
Library file:    REND<ENTER>
Operations:      -+OLV;<ENTER>
```

Ennek hatására a LIB a REND.LIB nevű könyvtárban az "olv" modult az aktuális tartalomjegyzékben található OLV.OBJ modullal helyettesíti. A ; jelzi, hogy a lista és az output file-ra az alapfeltételezés érvényes, vagyis keresztreferencia-lista nem készül, és a módosítás után az új könyvtárnak lesz REND.LIB a neve. (A régi REND.BAK néven megőrződik.)

```
LIB
Library file:    REND<Enter>
Operations:      -+OLV<ENTER>
List file:       REND.LST<ENTER>
Output file:     REND1.LIB<ENTER>
```

A LIB promptjaira adott válaszokkal a REND.LIB módosításával létrehozzuk a REND1.LIB könyvtárat, amelyben már az "olv" modul új változata van. Az új könyvtárról REND.LST néven készül keresztreferencia-lista.

7.2.3 LIB parancs-sorozat használata file-ból (válasz-file)

A LIB működése úgy is vezérelhető, hogy az előző pontban leírt, a promptokra adandó válaszokat előre elhelyezzük egy file-ban, az úgynevezett válasz-file-ban. A válasz-file-t szövegszerkesztő programmal (például az operációs rendszerhez adott EDLIN-nel) hozhatjuk létre. E file azonosítója tetszőleges lehet.

A válasz-file felépítése:

könyvtárnév [/**PAGESIZE**:*n*]
[*parancslista*]
[*lista-file*]
[*output-file*]

Így, rovatokra tagolva. Csak a parancs-listának lehet folytatássora, ilyenkor sorvégi & jelzi, hogy folytatás-sor következik. A ; jel itt is alkalmazható az előzőekben ismertetett módon.

A válasz-file aktivizálása:

LIB *@válasz-file azonosító*

Ennek hatására a LIB elindul. Működés közben a LIB kiírja a képernyőre a válasz-file feldolgozott sorait.

Ha a válasz-file-ban valamely kérdésre nem talál kielégítő választ, kiírja a kérdést a képernyőre, és feleletre vár.

Példa:

Az előző pontban ismertetett egyik párbeszédes példa összeállítása egy válasz-file-ban:

```
szamol.LIB
 + osszead + szoroz + kivon + oszt
szamol1.LIB
```

7.3 Új könyvtár létrehozása

A parancs formája:

LIB *könyvtárnév* [**/PAGESIZE:***n*]

Mint láttuk, ha egy létező könyvtár azonosítóját adtuk meg, a LIB úgy tekinti, hogy azt kívánjuk módosítani. Ha valóban új könyvtárat adtunk meg, akkor a LIB a képernyőre kiírja:

```
Library file does not exist. Create?
```

A válasz YES, ha létre akarunk hozni új könyvtárat, egyébként NO. A /PAGESIZE opció a könyvtár lapméretét határozza meg, amely alapértelmezés szerint 16 byte.

7.4 A lapméret átállítása (PAGESIZE)

A lapméret kettő hatvány 16 és 32768 között, az alapértelmezés 16. Ez az egész szám a lapméretet byte-okban adja meg.

A modulok a könyvtárban laphatáron kezdődnek a könyvtár elejétől számítva. Másrészt a LIB a program-modulok tárolásakor és viszszakeresésekor olyan indexelést használ, hogy nagyobb lapméreten több modul fér el. E két szempont figyelembevételével kell a lapméretet megválasztanunk úgy, hogy a helykihasználás, - figyelembe véve a könyvtárban elhelyezendő modulok méretét - optimális legyen.

Általában a kis méret az előnyös, kivéve, ha könyvtárunkban sok modul lesz.

A lapméret átállítása történhet parancs-sorral:

LIB *könyvtárnév* **/PAGESIZE:***n*

vagy a "Library name" prompt-ra adott válaszban:

/PAGESIZE:*n*

beírással a könyvtár nevének megadása után.

Az opció nevét röviden

/P:*n*

alakban is megadhatjuk.

7.5 A könyvtár konziszterciájának ellenőrzése

A könyvtár akkor konzisztens, ha a LIB könyvtár-nyilvántartása szerint a könyvtárban lévő programokat el is tudja érni, azaz "megvan az összes modul". Ilyen ellenőrzésre szükség lehet pl. egy lemez vagy lemezrész átmásolása után. Amikor egy könyvtárat készítünk egy vagy több modullal, a LIB automatikusan ellenőrzi a modulok konzisztenciáját. Az ellenőrzési funkció parancssor formájában:

LIB *könyvtárnév*;

interaktív üzemmódban a "Library name" kérdésre adott

könyvtárnév;

válasszal érhető el.

7.6 A keresztreferencia-lista és készítése

A keresztreferencia-lista valójában két listából áll: az egyik a könyvtárban lévő közösen használható (PUBLIC) szimbólumokat és az azokra hivatkozó modulneveket, a másik a könyvtárat alkotó modulok neveit és a bennük levő PUBLIC szimbólumokat tartalmazza.

Keresztreferencia-lista akkor készül, ha a LIB parancssorban való felhasználásakor megadjuk a "lista-file"-t, illetve interaktív üzemmódban a "List file" kérdésre adunk egy file-azonosítót.

8. Futtatható program előállítása - programszerkesztés

8.1 A LINK indítása

A LINK indítására és vezérlésére három lehetőség van:

- PC DOS parancs-sorral
- a LINK kulcsszóval, majd a LINK által kiírt kérdésekre adott válaszokkal
- egy előre összeállított válasz-file-lal.

A LINK működése bármikor leállítható a <Ctrl-Break> lenyomásával. A LINK a megadott tárgy-file-okból, valamint az esetleg szükséges könyvtári file-okból egy relokálható programot készít, amelynek helyigénye legfeljebb 1 Mbyte lehet.

A továbbiakban részletesen leírjuk, hogyan vezérelhető a LINK, és hogyan készíti el a futtatható programot.

8.1.1 Programszerkesztés DOS parancs-sorral

A szerkesztést indító parancs alakja:

LINK [*opciók*]*tárgy-file-ok*,[*futtatható file*],[*map-file*],[*könyvtár-file-ok*]

Az "*opciók*" a LINK működését vezérlő feltételek. Ezek részletes leírása e fejezet 2. pontjában található. A "*tárgy-file*" egy vagy több olyan tárgy-file (.OBJ) neve, amelyből a programot szerkesztjük. Ezeket a tárgy-file-okat valamelyik fordítóprogram - pl. makró-assembler, PASCAL vagy C - készítette. A "*futtatható file*" a LINK által elkészítendő program-file neve. Ha nem adjuk meg, akkor a LINK az első tárgy-file nevét feltételezi futtatható file-névként .EXE kiterjesztéssel. A "*map-file*"-ba kerül az elkészült program térképének listája. Ha nem nevezünk meg ilyen file-t, vagy az opciók között nem adjuk meg a /MAP vagy a /LINENUMBERS opciót, akkor nem készül map-file.

A "*könyvtár-file*" egy vagy több könyvtár-file neve, amely a programhoz szerkesztendő könytári rutinokat tartalmazza. Ha ilyenre nincs szükség, akkor a név helyébe pontosvesszőt kell írni.

A különböző típusú file-neveket vesszővel választjuk el egymástól. A vesszők kiírása akkor is kötelező, ha valamely file nevét nem adjuk meg. Ha egynél több tárgy-file vagy könyvtár-file neve szerepel, akkor a file-neveket legalább egy szóközzel vagy egy " + " jellel kell elválasztani egymástól. Az opciók bárhol lehetnek a parancs-sorban, csak azonos típusú file-nevek között nem.

A parancs-sorban szereplő file-nevekben nem kötelező megadni a kiterjesztéseket, mert a LINK feltételezi rendre az .OBJ (tárgy), .EXE (futtatható), .MAP (térkép) és .LIB (könyvtár) kiterjesztéseket.

Példák:

```
LINK UJIT.OBJ,UJIT.EXE,UJIT.MAP,UJIT.LIB
```

A parancs hatására a programszerkesztő az UJIT.OBJ tárgy-file-ból az UJIT.EXE végrehajtható program-file-t állítja elő az UJIT.LIB könyvtár-file-ban lévő megfelelő rutinok hozzászerkesztésével. A program térképe az UJIT.MAP file-ba kerül.

```
LINK UJIT,UJIT,UJIT,UJIT
```

A parancs hatása ugyanaz, mint az előzőé, mert a LINK automatikusan hozzáteszi a filenevekhez az .OBJ, .EXE, .MAP, .LIB kiterjesztést.

```
LINK UJIT,,UJIT,UJIT
```

A szerkesztés eredménye a LINK alapfeltételezései miatt ugyanaz, mint az előző két esetben.

```
LINK INIT + UJIT,UJIT,UJIT,;
```

Két tárgy-file-ból (INIT.OBJ és UJIT.OBJ) szerkeszti az UJIT.EXE file-t könyvtár használata nélkül, és elkészíti az UJIT.MAP file-t.

```
LINK X + Y + INIT/PAUSE,,XY,\KAT\ELEM
```

A LIB program az X.OBJ, Y.OBJ és INIT.OBJ tárgy-file-okból, valamint a "KAT" altartalomjegyzékben lévő ELEM.LIB könyvtár megfelelő rutinjaiból szerkeszti az X.EXE file-t és elkészíti az XY.MAP file-t. A /PAUSE opció hatására mielőtt a LINK elkezdené a futtatható program előállítását, a végrehajtás felfüggesztődik.

8.1.2 Programszerkesztés interaktív üzemmódban

Lehetőség van arra is, hogy a programszerkesztést párbeszédes üzemmódban hajtsuk végre. Ilyenkor az előző pontban ismertetett parancslista elemeit részenként adjuk meg.

A párbeszédes programszerkesztés lépései:

1. Begépeljük a szerkesztő program nevét:

```
LINK
```

és megnyomjuk az <Enter> gombot. Ennek hatására a LINK megkérdezi a tárgymodulok nevét. A képernyőn a következő üzenet jelenik meg:

```
Object Modules [.OBJ]:
```

2. Az előbbi üzenetre válaszul beírjuk az összeszerkesztendő tárgy-file-ok nevét. Egynél több tárgy-file esetén a file-neveket szóközökkel vagy "+" jellel kell elválasztani egymástól. Ha a file-nevek nem férnek el egy sorban, akkor a sor utolsó karaktereként a "+"-t üssük le, majd nyomjuk meg az <Enter> gombot! Ekkor a LINK megkérdezi a többi tárgy-file nevét. A file-nevekben nem kötelező kiírni a .OBJ kiterjesztést, mert ez az alapfeltételezés.

Ha megadtuk az összes tárgy-file nevét, nyomjuk meg az <Enter>-t! Ennek hatására a LINK a futtatható file nevét kéri:

Run File [file-név.EXE]:

3. Beírjuk a futtatható file nevét és megnyomjuk az <Enter>-t. Ha nem adunk meg file-nevet vagy kiterjesztést, akkor a LINK az alapértelmezést veszi figyelembe (első tárgymodul neve .EXE illetve .EXE). Ilyenkor elég csak az <Enter>-t megnyomni.

A LINK ezután a map-file nevét kérdezi:

List File [NUL.MAP]:

4. Ha készíteni akarunk map-file-t, akkor írjuk be a nevét és nyomjuk meg az <Enter>-t! A .MAP kiterjesztést nem kötelező megadni, mert alapfeltételezés. Ha nem kérünk map-file-t, akkor elegendő csak az <Enter>-t megnyomni. A LINK következő kérdése a könyvtár neve:

Libraries [.LIB]:

5. Valamennyi könyvtár nevét fel kell sorolni, amelyekben olyan rutinok vagy változók vannak, amelyekre a program hivatkozik, de nem deklarálja azokat. A nevek elválasztása szóközökkel vagy "+" jellel történik. A kiterjesztés alapértelmezése .LIB. A könyvtárneveket több sorban is megadhatjuk úgy, mint a tárgy-file-ok neveit. Ha a könyvtarokat valamelyik fordító automatikusan jelzi a LINK-nek, de ezek nem érhetők el az aktuális útvonal(ak)on, elég a LINK-nek megadni azt, hogy a könyvtárak melyik altartalomjegyzékben vannak. Ilyenkor:

[drive:]\út\

alakú választ kell adnunk. (Pl: C:\COMP\MSC\)

Ha könyvtárhasználatra nincs szükség, akkor elég csak az <Enter>-t leütni.

Ezek után a LINK előállítja a végrehajtható file-t.

Megjegyzések:

Ha valamely file másik könyvtárban vagy másik lemezen van, mint az aktuális alapértelmezés, akkor a file nevéhez meg kell adni a megfelelő elérési út (path) vagy lemez nevét. Ha a LINK nem talál meg egy tárgy-file-t, akkor üzen és vár, amíg lemezt cserélünk, ha szükséges.

Minden sor végén lehetőség van opciók megadására.

Bármely üzenetváltás végződhet pontosvesszővel, amelynek hatása: a LINK befejezi a párbeszédet, az elhagyott kérdésekre az alapfeltételezést alkalmazza és elvégzi a programszerkesztést. Ügyeljünk arra, hogy legalább egy tárgy-file neve szerepeljen!

Példa:

Az 1.1 pontban leírt utolsó példa párbeszédes formája a következő:

```
LINK
Object Modules [.OBJ]: x+y+
Object Modules [.OBJ]: init/PAUSE
Run File [x.EXE]:
List File [NUL.MAP]:xy
Libraries [.LIB]:\kat\elem
```

Az első sort a felhasználó írja. A 2. és 3. sor példa a file-neveknek több sorban történő megadására. A 4. sorban láthatjuk a név alapfeltételezését (x), amely a végleges is, mert nem adtunk másikat.

8.1.3 Programszerkesztés előre összeállított válasz-file-lal

A LINK működéséhez szükséges információkat előre elhelyezhetjük egy file-ban, amelyet válasz-file-nak nevezünk. Ekkor a szerkesztés indítása:

LINK @válasz-file-név

A válasz-file neve tetszőleges lehet. A válasz-file tartalma:

tárgy-file-név
[futtatható file-név]
[map-file-név]
[könyvtár-file-nevek]

A file-ok megadási módjára az előzőekben leírt szabályok érvényesek. Minden file-típusnak külön sorban kell szerepelnie. Ha valamelyik file-típus helyett az alapfeltételezést használjuk, akkor a megfelelő sort üresen kell hagyni.

Indítás után a LINK elővezi a válasz-file-ból a megfelelő információkat és automatikusan elvégzi a programszerkesztést. Szerkesztés közben a válasz-file tartalma folyamatosan megjelenik a képernyőn. Ha a LINK hibát talál a nevek olvasásakor, hibajelzést ad és megvárja, amíg beírjuk a helyes választ.

Példa:

Az előző pontban bemutatott párbeszédes programszerkesztési példának megfelelő válasz-file a következő:

```
x+y+
init/PAUSE

xy
\kat\elem
```

Figyeljük meg, hogy a harmadik sor üres! Ha az xy ott volna, akkor azt a LINK a Run File kérdésre adott válasznak tekintené.

8.1.4 Könyvtárak elérési útjainak megadása

A könyvtárak elérési útjainak megadására két módszer lehetséges:

Az első esetben a LINK parancsban lemezegységek és alkönyvtárak felsorolásával megadhatjuk azokat az elérési utakat, amelyeken haladni kell az adott név keresésekor.

A másik lehetőség az, hogy a LINK hívása előtt a PC DOS SET parancsával értéket adunk a LIB környezetleíró változónak. Ez az érték egy vagy több elérési út. Legfeljebb 16 elérési utat adhatunk meg. Az egyes elérési utakat pontosvesszővel választjuk el egymástól. Egy elérési út leírása:

lemezegység:\altartalomjegyzék

pl. A:\katal1

Ha a lemezegység vagy az altartalomjegyzék hiányzik, akkor az aktuális lemezegység illetve altartalomjegyzék a feltételezés.

Több elérési út megadása esetén a LINK a felírás sorrendjében balról jobbra haladva keresi a megfelelő könyvtárat. Ha például a kt.LIB könyvtár elérési útjai:

```
A:\kat1;\kat2;B:
```

akkor először az "A" lemezegység "kat1" altartalomjegyzékében keresi kt.LIB-et. Ha nem találja, ugyanezen a lemezegységen a "kat2" altartalomjegyzék következik, s ha abban sincs, áttér a "B" lemezegységre és annak az aktuális altartalomjegyzékben folytatja a keresést.

Példák:

```
LINK rend,,rend.MAP,A:\kat1\kt.LIB + dcl + B: + C:\kat2\
```

A LINK a "kt.LIB" könyvtárat csak az "A" lemezegység "kat1" altartalomjegyzékben keresi, mert közvetlenül a könyvtárnév előtt adtuk meg az elérési utat. A "dcl" könyvtárat először az "A" lemezegység aktuális altartalomjegyzékében keresi, majd a "B" lemezegység aktuális altartalomjegyzékében, végül a "C" lemezegység "kat2" altartalomjegyzékében.

```
SET LIB = A:\kat1;B:;C:\kat2
LINK rend,,rend.MAP,kt + dcl
```

Ebben a példában a LINK először az aktuális lemezegység aktuális altartalomjegyzékében keresi a "kt" és a "dcl" könyvtárat. Ha nem találja, akkor az "A" egység "kat1" altartalomjegyzékében, majd a "B" egység aktuális altartalomjegyzékében, végül a "C" egység "kat2" altartalomjegyzékében folytatja a keresést.

8.1.5 A map-file szerkezete

A programról készített map-file lényegében a program főbb részeinek a program kezdetéhez relatív elhelyezéséről ad információt, valamint jelzi az esetleges hibákat.

Ha a LINK parancsban megadtuk a /MAP opciót, akkor kilistázza a public szimbólumokat és azok betöltési címeit is.

A map-file-ban lévő szegmens-információ alakja:

```
START     STOP      LENGTH    NAME    CLASS
kkkkkH    vvvvvH    lllllH    x...x   y...y
  .         .         .         .       .
  .         .         .         .       .
  .         .         .         .       .
```

A táblázat első két oszlopa a szegmensek kezdő- és végcímét mutatja 20 bites hexadecimális alakban. Ezek a címek a load modul kezdetéhez relatívek. A load modul feltételezett kezdőcíme 00000H. Ezekből a címekből képezi az operációs rendszer a valódi címeket a program betöltésekor. A harmadik oszlop a szegmens byte-okban mért hosszát adja meg. A negyedik oszlopban van a szegmens neve, az ötödikben a szegmens típusa (pl. CODE-utasítások, DATA-adatok).

A map-file-ban lévő csoport-információ általános alakja:

```
ORIGIN    GROUP
nnnn:n    x...x
  .         .
  .         .
  .         .
```

Az első oszlop a csoport kezdetét, a második a csoport nevét mutatja.

A LINK a program belépési pontjának címét a map-file végén helyezi el.

Ha a LINK parancssorában megadjuk a /MAP opciót, akkor a LINK a public szimbólumokról is elhelyez két táblázatot a map-file-ban. Az egyik táblázatban abc-sorrendben, a másikban betöltési cím szerinti sorrendben szerepelnek a szimbólumok. A címek szegmens:relatív cím alakban kerülnek a map-file-ba. Ezek a load modul kezdetéhez relatív címek. A load modul feltételezett kezdőcíme 00000H.

Ha a LINK parancsban a /HIGH és /DSALLOCATE opció is szerepel és a program mérete az adatokkal együtt nem haladja meg a 64 Kbyte-ot, akkor az előzőekben említett táblázatokban egyes szimbólumok mellett meglepően nagy címet találhatunk. Ez azt jelenti, hogy a szimbólum a program elé kerül.

8.1.6 Ideiglenes lemez-file - VM.TMP

A LINK munkaterülete a memóriában van. Ha ez a terület programszerkesztés közben megtelik, akkor a LINK létrehoz egy ideiglenes lemez-file-t, amelynek neve az aktuális alkatalomjegyzékben VM.TMP lesz. Ekkor a LINK a következő üzenetet küldi:

```
VM.TMP has been created.
Do not change diskette in drive, <d:>
```

Az üzenet jelentése:

> VM.TMP készült.
> Ne cserélje a lemezt a < d: > egységen!

Ez az üzenet a LINK működése végéig érvényes. A szerkesztés végrehajtása után az ideiglenes lemez-file automatikusan törlődik. *Ha a LINK ideiglenes file-t hoz létre, az esetleg létező VM.TMP nevű file-t törli. Ezért ezt a nevet saját célra ne használjuk!*

8.2 A LINK opciói

Az opciók segítségével vezérelhetjük a LINK működését. Minden opció a "/" jellel kezdődik és a LINK parancsban bárhol előfordulhat, csak azonos típusú file-nevek között nem. A gyorsabb írás kedvéért minden opció neve rövidíthető.

Az opciók listája:

/PAUSE	megszakítja a szerkesztést
/MAP	kiírja a public szimbólumok tábláját
/STACK	verem-méret
/CPARMAXALLOC	a helyigény maximuma a tárban
/HIGH	betöltés a lehetséges legnagyobb címtől
/DSALLOCATE	helyfoglalás adatcsoportnak
/OVERLAYINTERRUPT	megszakítás overlay esetén
/LINENUMBERS	forrás-sorok kezdőcímének kiírása
/NOIGNORECASE	kis- és nagybetűk megkülönböztetése
/NOGROUPASSOCIATION	csoport-hozzárendelés törlése
/NODEFAULTLIBRARYSEARCH	könyvtár-alapfeltételezés törlése
/SEGMENTS	szegmensek számának maximuma
/DOSSEG	MS-DOS szegmens-elrendezés használata

8.2.1 Megszakítás a szerkesztés alatt - PAUSE

Alakja:

> **/PAUSE** rövidítve: **/P**

Ezzel az opcióval megszakítható a LINK működése, mielőtt a végrehajtható file-t (.EXE) lemezre írja. Akkor szükséges, ha a kiírás előtt lemezt akarunk cserélni. Ekkor a LINK üzenete:

```
About to generate .EXE file
Change disks < hit any key >
```

Az üzenet jelentése:

A .EXE file generálásához cserélje ki a lemez(eke)t és nyomja meg valamelyik billentyűt! Egy billentyű lenyomása után a LINK folytatja munkáját. *Ha VM.TMP nevű file van már a lemezen, akkor ezt a lemezt nem szabad cserélni.*

Példa:

```
LINK rend/PAUSE,,,\kat\elem
```

A parancs hatása az, hogy a REND.EXE file lemezre írása előtt a LINK megáll, majd a lemezcsere után folytatja a működést. A REND.EXE file lemezre írása után ismét megáll, lehetővé téve az eredeti lemez visszahelyezését.

8.2.2 Public szimbólumok táblájának előállítása - MAP

Alakja:

/MAP rövidítve: /M

Az opció hatására a LINK elkészíti a public szimbólumok táblázatát abc-sorrendben, valamint cím szerint növekvő sorrendben (ld. 8.1.5) és elhelyezi a map-file-ban. Ha a LINK parancsban nem jelöltünk ki map-file-t, akkor a lista-file előtt megadott /MAP opció hatására automatikusan készül ilyen file, amelynek neve az első tárgy-file neve .MAP kiterjesztéssel.

Példa:

```
LINK rend/MAP,;
```

ami egyenértékű a

```
LINK rend.OBJ,rend.EXE,rend.MAP/MAP,;
```

paranccsal.

8.2.3 A verem-méret meghatározása - STACK

Alakja:

/STACK:*méret* rövidítve: /ST:*méret*

Hatására a programhoz tartozó verem helyfoglalása "méret" byte lesz. Értéke

1 < = méret < = 65535

lehet. A méret megadható decimális, oktális vagy hexadecimális formában. Az oktális formának 0-val (nullával) kell kezdődnie, a hexadecimális formának 0X-szel.

A LINK általában automatikusan kiszámítja a verem méretét a tárgymodulokban megadott verem-szegmensek méretei alapján. Ezt a számított értéket lehet felülbírálni a LINK parancs /ST opciójával. Ha nincs verem-szegmens a programunkban, akkor a LINK hibajelzést ad. Ezt elkerülendő, minden programban legalább egy vermet definiáljunk!

Példa:

```
LINK rend/ST:512,;
```

vagy más alakban

```
LINK rend.OBJ/STACK:512,rend.EXE;
```

Mindkét parancs 512 byte helyet biztosít a verem számára a "rend" nevű programban (futtatáskor). Az 512-t megadhattuk volna oktális (01000) vagy hexadecimális (0X200) alakban is.

(Megjegyzés: Ha egy programból .COM kiterjesztésű futtatható file-t akarunk készíteni, nem szabad vermet definiálni benne. Ilyenkor a LINK természetesen hibaüzenetet ad, de utána a .EXE file-t az EXE2BIN programmal átalakíthatjuk .COM formátumra.)

8.2.4 A program helyigényének maximuma - CPARMAXALLOC

A program maximális helyigényét a CPARMAXALLOC opcióval határozhatjuk meg, amelynek alakja:

/CPARMAXALLOC:*szám* rövidítve: /C:*szám*

Ennek hatására az operációs rendszer "szám" darab paragrafusnyi területet foglal le a memóriában, mielőtt betölti a programot. (Egy paragrafus hossza 16 byte.) Értéke

1 < = szám < = 65535

lehet. A "szám" megadható oktálisan és hexadecimálisan is. Ekkor 0-val, illetve 0X-szel kell kezdődnie.

Ha nem adjuk meg ezt az opciót, a LINK alapértelmezésben 65535-nek veszi a paragrafusok maximális számát. Mivel ez a teljes tár mérete, az operációs rendszer csak annyi helyet biztosít, amekkora a tár legnagyobb összefüggő blokkja.

Ha megadtuk a CPARMAXALLOC opciót, akkor a lefoglalt paragrafusok száma a "szám"-mal egyenlő, és a fennmaradó memóriaterületet más programok használhatják. Abban az esetben, ha kisebb számot adunk meg, mint amennyi helyre a betöltésre kerülő programnak szüksége van, a LINK automatikusan beállítja a helyigénynek megfelelő hosszúságot.

A program által igényelt paragrafusok számának minimuma sohasem kisebb, mint a program kód- és adatszegmensei paragrafusainak együttes száma.

Példa:

```
LINK rend/C:127,;
```

azaz

```
LINK rend.OBJ/CPARMAXALLOC:127,rend.EXE,;
```

ami 127 paragrafus (2032 byte) helyfoglalást jelent. A 127 helyett írhattuk volna:

oktálisan: 0177,
hexadecimálisan: 0X7F.

8.2.5 A kezdőcím maximumra állítása - HIGH

Az opció hatására a program kezdőcíme a lehető legnagyobb cím lesz a tár szabad területén. Az opció megadási módja:

/HIGH rövidítve: **/H**

Így lehet a programot a szabad hely végében elhelyezni. Ha nem adunk meg ilyen opciót, akkor a betöltés a szabad hely elején kezdődik. Az opció csak a DSALLOCATE paraméterrel együtt használható.

8.2.6 Helyfoglalás az adatcsoportnak - DSALLOCATE

Az opció hatására a DGROUP nevű csoportot tartalmazó szegmensben a csoport a lehető legnagyobb címre kerül. Az opció alakja:

/DSALLOCATE rövidítve: **/D**

A LINK alapértelmezésben az adatcsoport első byte-jának szegmens-relatív címét 0000H-nak adja meg, míg a /D opció hatására a csoport utolsó byte-jához az FFFFH relatív címet rendeli. Ezt az opciót legtöbbször a /HIGH opcióval együtt használjuk, mert így biztosítani tudjuk, hogy a program előtti fel nem használt területtel jobban gazdálkodjunk.

8.2.7 Forrás-sorok kezdőcímének kiírása - LINENUMBERS

Az opció hatására a LINK a program minden forrás-sorából képzett utasítás(ok) kezdőcímét a map file-ba írja. Az opció alakja:

/LINENUMBERS rövidítve: **/LI**

A LINK által így készített információt fel lehet használni magasabb szintű nyelven írt programok forrásszintű nyomkövetéssel való teszteléséhez. Az opció megadásának hatására csak akkor képez a LINK utasítás-kezdőcímeket, ha:

- van map-file-név a LINK parancsban
- a tárgy-file-ban van sorszámozás (csak a magasszintű nyelvek fordítói készítenek ilyet).

Ha nincs a tárgy-file-ban sorszámozás, akkor a LINK figyelmen kívül hagyja a LINENUMBERS opciót. Ha nem adtunk meg map-file-nevet, akkor a LINENUMBERS a MAP opció funkcióját is ellátja.

8.2.8 Kis- és nagybetűk megkülönböztetése - NOIGNORECASE

Az opció hatására a LINK különbözőknek tekinti a kis- és a nagybetűket. Alakja:

/NOIGNORECASE rövidítve: **/NOI**

Ezt az opciót olyankor használjuk, ha az összeszerkesztendő tárgy-file-ok között vannak olyanok, amelyekben a kis- és a nagybetűk különbözők. Ezeket a tárgy-file-okat olyan magasszintű nyelv fordítóprogramja (pl. C fordító) készítette, amely különbséget tesz a két betűtípus között.

Alapértelmezésben a LINK azonosnak tekinti a nagy- és kisbetűket, pl. MAP, Map és map azonos szimbólumok.

8.2.9 A könyvtár-alapfeltételezés törlése - NODEFAULTLIBRARYSEARCH

Alakja:

/NODEFAULTLIBRARYSEARCH rövidítve: /NOD

A magasszintű nyelvek fordítói a tárgy-file-hoz hozzátehetik azon könyvtárak nevét, amelyek használatát feltételezik. Ez az információ felülbírálható a LINK parancs /NOD opciójával és egy másik könyvtár megadásával.

Tegyük fel például, hogy a "rend" nevű rendezőprogram C nyelven készült és hivatkozunk benne egy C nyelvű programokat tartalmazó könyvtárra, amelyek a rendezéshez szükséges eljárások. Ezután macro-assembly-ben elkészítünk egy ugyanilyen funkciójú könyvtárat, felgyorsítva az eljárásokat. Ekkor a /NOD opcióval és az új könyvtár nevének megadásával a gyorsabb eljárások szerkeszthetők a programhoz.

8.2.10 Csoport eltávolítása a programból - NOGROUPASSOCIATION

Az opció hatására a LINK figyelmen kívül hagyja a csoportot, amikor a cím hozzárendeléseket végzi. Alakja:

/NOGROUPASSOCIATION rövidítve: /NOG

Lehetőleg ne használjuk!

8.2.11 Megszakítás overlay esetén - OVERLAYINTERRUPT

Alakja:

/OVERLEYINTERRUPT:*szám* rövidítve: /O:*szám*

Ezzel az opcióval beállíthatjuk az overlay betöltő rutin megszakítási számát, azaz felülbíráljuk az alapértelmezés szerinti megszakítási számot (03FH). Értéke

0 < = szám < = 255

lehet. A számot decimális, oktális vagy hexadecimális alakban adhatjuk meg. Az oktális számnak 0-val, a hexadecimálisnak 0X-szel kell kezdődnie.

Ügyeljünk arra, hogy ne használjunk olyan megszakítási számokat, amelyeket a hardver használ, illetve nem illeszkednek az operációs rendszer megszakításaihoz!

8.2.12 Szegmensek maximális számának meghatározása - SEGMENTS

Alakja:

/SEGMENTS:*szám* rövidítve: /SE:*szám*

Az opció hatására a LINK legfeljebb annyi szegmenst dolgoz fel a program szerkesztésekor, amennyi az itt megadott szám. Ha a program ennél több szegmensből áll, akkor a LINK hibajelzést ad és befejezi működését. Értéke

1 < = szám < = 1024

lehet. A maximális szegmensszám decimálisan, oktálisan vagy hexadecimálisan adható meg. Az oktális számnak 0-val, a hexadecimálisnak 0X-szel kell kezdődnie.

Ha a LINK parancsban nincs /SE opció, akkor 128 az alapfeltételezés, azaz a LINK 128 szegmensnek elegendő helyet foglal le a tárban.

8.2.13 Az MS-DOS szegmens-elrendezés alkalmazása - DOSSEG

Alakja:

/DOSSEG rövidítve: /DO

Az opció hatására a LINK az MS-DOS operációs rendszer szegmens-elrendezési konvencióinak megfelelően helyezi el a végrehajtható file szegmenseit.

Az MS-DOS szegmens-elrendezési szabályai:

- A "CODE" osztálynevű szegmensek a végrehajtható file elejére kerülnek.
- Közvetlenül a "CODE" szegmens után helyezi el azokat a szegmenseket, amelyek nem tartoznak a "DGROUP" csoportba.
- A "DGROUP"-hoz tartozó valamennyi szegmens a file végére kerül.

8.3 A LINK működése

A LINK a parancsban megadott megfelelő opciók és a forrás-file-okban lévő direktívák alapján elkészíti a végrehajtható file-t a programok kód- és adatszegmenseit összekapcsolva. Az így kapott "kép" (image) már a program végső formája, ami változatlanul kerül be a tárba, ha a programot futtatni akarjuk.

A LINK számára a szegmensek és a szegmens-csoportok összeszerkesztésének módját a szegmens-attributumokkal adhatjuk meg a forrás-szövegben, továbbá a LINK parancsban lévő információkkal. A SEGMENT direktíva a szegmensek összekapcsolásához, a GROUP direktíva a szegmens-csoportok kialakításához ad információt. Ezekkel a direktívákkal definiálhatjuk a program valamennyi szegmensének sorrendjét és relatív kezdőcímét. Az "elrendezés" típusai:

- csoportok összeállítása
- azonos típusú csoportok összegyűjtése (osztályok)

- címhatárra igazítás (alignment).

A továbbiakban leírjuk, hogy a LINK hogyan kapcsolja össze a szegmenseket, és hogyan oldja fel a hivatkozásokat.

8.3.1 A szegmensek címhatárra való igazítása (ALIGNMENT)

A szegmensben lévő információ típusának megfelelően megadhatjuk a kezdőcímre vonatkozó kikötésünket.

típus	megnevezés	feltétel a címre
BYTE	tetszés szerinti byte címe	egész szám
WORD	egy szó címe	2-vel osztható szám
PARA	egy paragrafus címe	16-tal osztható szám
PAGE	egy lap címe	1024-gyel osztható szám

Mielőtt a LINK egy szegmenst a végrehajtható file-ba másol, megvizsgálja a soronkövetkező byte címét, hogy kielégíti-e a kezdőcím típusának megfelelő feltételt. Ha igen, akkor folyamatosan kimásolja a szegmenst. Ha nem, akkor a feltételnek megfelelő következő határig üres helyet hagy és azután kezdi másolni a szegmenst.

8.3.2 A szegmensek kezdőcímének meghatározása

Amikor a LINK összeállítja a végrehajtható file-t, a soronkövetkező szegmens lemezre írása előtt kiszámítja a szegmens program-relatív kezdőcímét. Ezt a címet a már kiírt szegmensek együttes hossza és a szegmens kezdőcímének típusa határozza meg. A cím két részből áll:

- paragrafuscím (frame number): az első olyan paragrafus program-relatív címe (16-tal osztható), amely a szegmens elejét tartalmazza
- eltolás: a szegmens első byte-jának távolsága a paragrafus kezdetétől.

Az eltolás értéke csak akkor lehet nullától különböző, ha a szegmens kezdőcíme BYTE vagy WORD típusú. A szerkesztésről kért listában a szegmens kezdőcímének első 5 hexadecimális jegye a frame number.

8.3.3 A szegmensek átrendezése

A LINK olyan sorrendben helyezi el a szegmenseket a végrehajtható file-ban, ahogyan felsoroltuk a tárgy-file-okat a LINK parancsban.

Ha azonos osztálynevű szegmenseket talál, akkor azokat egy folyamatos területen helyezi el, így eltérhet a tárgy-file-ok megadási sorrendjétől.

Abban az esetben, ha a szegmens osztályát nem definiáltuk, a "nulla" osztálynév rendelődik a szegmenshez. Ha a LINK több ilyet talál, ezeket is egy folyamatos területre teszi. Az azonos osztályba tartozó szegmensek száma és együttes helyfoglalásuk nem korlátozott. Az utóbbi meghaladhatja a 64 Kbyte-ot.

8.3.4 Kombinált szegmensek

A szegmens attributumai közöt a forrás-programban a következő kombinációs típusok adhatók meg:

PUBLIC, STACK, MEMORY, COMMON, PRIVATE.

A LINK az azonos nevű és kombinációs típusú szegmenseket egyetlen nagy szegmensbe gyűjti, amelynek mérete legfeljebb 64 Kbyte lehet. Túllépéskor a LINK hibajelzést ad.

A PUBLIC kombinációs típusú, azonos nevű szegmensek előfordulási sorrendben való egymás utáni elhelyezésével egyetlen nagy szegmens készül. Az így kapott szegmens valamennyi elemének címe ugyanahhoz a "frame address"-hez relatív. A LINK megőrzi az egyes szegmensek igazítási attributumát, és a szegmensek bemásolásánál figyelembe veszi.

A STACK vagy MEMORY típusú szegmensek kombinálása a PUBLIC típusúaknak megfelelően történik, de a STACK esetében a mutató kezdőértéke is a végrehajtható file-ba kerül.

A COMMON típusú szegmensek kombinációjának feltétele: azonos szegmensnév és azonos osztálynév. A kombinált szegmens hossza megegyezik a leghosszabb összetevő szegmens hosszával. Valamennyi összetevő szegmens kezdőcíme egyenlő a kombinált szegmens kezdőcímével, így átlapolva biztosítható a közös terület létrehozása.

Ha a leírt négy kombinációs típus egyike sem szerepel, akkor a szegmenst PRIVATE típusúnak nevezzük. Az ilyen szegmensekből nem készül kombinációs szegmens.

8.3.5 Szegmens-csoportok

Különböző osztályokhoz tartozó és különböző kombinációs típusú szegmensekből szegmens-csoport készíthető, amelynek mérete nem haladhatja meg a 64 Kbyte-ot. Ennek előnye, hogy a csoport szegmens-címe valamennyi benne lévő szegmens közös bázisa lesz, amelyhez relatív a csoportban lévő valamennyi cím. A csoport szegmenseinek elhelyezése nem feltétlenül folytonos, lehetnek közöttük más szegmensek is, de az összméret értelemszerűen ezekkel együtt sem haladhatja meg a 64 Kbyte-ot.

8.3.6 Címfeloldás

Ha a LINK a program valamennyi szegmensének kezdőcímét ismeri, akkor feloldja a még definiálatlan címke- és változó-hivatkozásokat. A már korábban elkészített relatív cím és a kezdőcím adja a valódi, program-relatív címet. A címfeloldás négy különböző hivatkozásra vonatkozik:

- rövid
- közeli önrelatív
- távoli szegmens-relatív
- hosszú.

A kiszámított érték mérete a hivatkozás típusától függ. Ha a megengedett mérettől eltérés van, hibajelzést kapunk. Ez például akkor történhet meg, ha egy 16-bites relatív címet akarunk használni egy olyan szegmensben lévő objektum eléréséhez, amely más "frame address"-hez relatív. Akkor is hibajelzést kapunk, ha valamely szegmenscsoporthoz tartozó szegmensek együttes mérete meghaladja a 64 Kbyte-ot.

A rövid hivatkozás - például a JMP utasításban - olyan címkézett utasításra vonatkozik, amely ugyanabban a szegmensben vagy szegmens-csoportban van. Egy ilyen hivatkozás legfeljebb 128 byte távolságra történhet. A relatív cím 8-bites lesz. Ha az előbbi feltételek nem teljesülnek, akkor hibajelzést kapunk.

A közeli önrelatív hivatkozás olyan adatra vonatkozik, amely ugyanabban a szegmensben vagy szegmens-csoportban van. Ez a relatív cím 16-bites. Egyéb hivatkozáskor hibajelzést kapunk.

A távoli szegmens-relatív hivatkozás olyan adatra vonatkozik, amely egy meghatározott szegmensben vagy szegmens-csoportban van, vagy egy adott szegmensregiszterhez relatív. Ez a relatív cím 16-bites. Egyéb hivatkozáskor hibajelzést kapunk.

Hosszú hivatkozás a JMP vagy CALL utasításban fordulhat elő, amellyel egy másik szegmensben vagy szegmens-csoportban lévő utasításra hivatkozunk. A LINK ekkor kiszámítja a 16-bites "frame address"-t és a 16-bites relatív címet. Ha a cím nem számítható, hibajelzést kapunk.

8.3.7 A betöltés sorrendjének vezérlése

A szegmensek alapfeltételezés szerinti betöltési sorrendjétől úgy lehet eltérni, hogy írunk egy álprogram-file-t (dummy), benne a megfelelő ál-szegmensekkel, amelyek osztálynév szerinti sorrendje megegyezik a betöltés kívánt sorrendjével. Az álprogram lefordítása után a kapott tárgy-file-t kell megadni a LINK parancsban első tárgy-file-ként. Ekkor a LINK automatikusan a leírt sorrendben tölti be a megfelelő szegmenseket.

Ha pl. egy programban a szegmensek nevei a következők:

kód, adat, verem, konstans, tár

akkor egy adott sorrendet biztosító álprogram-file felépítése:

```
kod        SEGMENT    'CODE'
kod        ENDS
konstans   SEGMENT    'CONST'
konstans   ENDS
adat       SEGMENT    'DATA'
adat       ENDS
verem      SEGMENT    'STACK'
verem      ENDS
tar        SEGMENT    'MEMORY'
tar        ENDS
```

Az álprogramban minden olyan osztálynevű szegmensnek szerepelnie kell, amilyen a szerkesztésre váró programban előfordul. Ha a felsorolás hiányos, a szegmensek sorrendje eltérhet a kívánt sorrendből. Egy program szerkesztésekor a LINK parancsban megadott első tárgy-file az a file, amelyet az álprogramból készítettünk.

Megjegyzések:

- A magasszintű nyelveken írt programokban a szegmensek sorrendjét a megfelelő rendszer határozza meg, ezért az álprogram használatának nincs értelme.
- Ha a MEMORY szegmenseket a program végére akarjuk szerkeszteni, akkor az álprogram végére egy üres MEMORY szegmenst írjunk:

```
tar SEGMENT MEMORY 'MEMORY'
tar ENDS
```

9. Programok karbantartása a MAKE program segítségével

9.1 Bevezetés

Programok fejlesztése során mindenkinek szembe kell néznie a programok "karbantartásának" problémájával. Nagyobb programok készítésekor elkerülhetetlen, hogy több modulból építkezzünk, viszont ez igen sok kellemetlenséget is okozhat.

Vegyük példaként az alábbi esetet: Programunkat a következő modulokból építettük fel: PROG.ASM, P1.ASM, P2.ASM (vagy más nyelvek esetén .C, .PAS, stb). A futtatható program neve legyen PROG.EXE.

Ha a forrásfile-ok valamelyikén változtatunk, akkor azt le is kell fordítanunk és linkelnünk, hogy az új futtatható file létrejöjjön. Gyakori eset, hogy több forrásfile-t is módosítunk, de elfelejtjük mindet lefordítani. A futó programba így a régi object file szerkesztődik be, és a programunk továbbra sem működik, pedig mi "kijavítottuk".

Ezt a hibát kiküszöböli és egyben sok fáradtságtól is megkímél bennünket a Microsoft MAKE segédprogramja. Segítségével automatizálható a program "karbantartása", azaz a fordítás, könyvtárazás, szerkesztés. A továbbiakban a MAKE program 4.02-es verziójának használatát és tulajdonságait részletezzük.

9.2 A MAKE használata

Először létre kell hoznunk a MAKE számára egy ún. leíró file-t. Ezt tekinthetjük egy speciális BATCH file-nak is, melyet nem a DOS, hanem a MAKE program fog feldolgozni. A leíró file általában feltételeket és parancsokat tartalmaz az alábbi formában:

```
output-file: input-file1 [input-file2] ...
parancs1
[parancs2]
...
```

E sorok jelentése a következő: Ha az input file-ok valamelyikének létrehozási időpontja későbbi, mint az output file-é (azaz az input file-ok valamelyike megváltozott az output file létrehozása óta), végre kell hajtani a parancso(ka)t.

Amennyiben a leíró file-unk az alábbi sorokat tartalmazza,

```
PROG.EXE: PROG.OBJ P1.OBJ P2.OBJ PROG.LIB
LINK PROG + P1 + P2,,,PROG ;
```

a program újraszerkesztésére csak akkor kerül sor, ha a legutóbbi szerkesztés óta megváltozott a PROG.OBJ, P1.OBJ, P2.OBJ, PROG.LIB file-ok valamelyike.

9.3 MAKE leíró file létrehozása

A file-t bármely szövegszerkesztővel létrehozhatjuk. A file-név kiterjesztése bármi lehet, általában elhagyjuk.

A leíró file-ban minden feltétel és parancs külön sorba kell kerüljön. Ha a feltétel túl hosszú, akkor azt egy backslash (\) után új sorban folytathatjuk.

A megadott input file-oknak létezniük kell, ellenkező esetben a MAKE nem tudja értelmezni a feltételt és végrehajtani a parancsokat. Az output file-nak nem kell léteznie, hiszen azt például a következő parancs is létrehozhatja.

A megadott parancs tetszőleges, a DOS számára felismerhető parancs lehet.

Az egyes input file-okat (legalább egy) szóköz karakterrel kell elválasztanunk. Az egyes parancsokat egy TAB vagy szóköz előzze meg, és mindegyiket új sorba kell írnunk.

A leíró file-ban több "leírás" (feltétel-parancs pár) is lehet. Az egyes leírásokat legalább egy üres sor válassza el!

A leíró file a fentieken kívül tartalmazhat még megjegyzéseket is. Ezeket mindig a # karakternek kell megelőznie. A megjegyzés kerülhet az input file-ok vagy a parancsok után, vagy egy új sor legelejére (szóköz nem előzheti meg!).

Ez utóbbi esetben a sor csak a megjegyzést tartalmazhatja.

9.4 A MAKE futtatása

A program indítása a következő:

MAKE [*opciók*] [*makródefiníciók*] *leírófile*

Például ha létrehoztuk PROG.EXE programunk karbantartására a PROG.MAK nevű leírófile-t, azt a következőképpen indíthatjuk el:

```
MAKE PROG.MAK
```

Ennek hatására a MAKE a PROG.MAK-ban lévő leírásokat sorra veszi, s a leírt feltételek teljesülésétől függően az egyes parancsokat végrehajtja. Végrehajtás során a MAKE felismeri, a DOS környezet-változóit (pl. PATH), s ezt, pl. a file-nevek megadásakor figyelembe vehetjük (azaz a teljes útvonalnevet csak akkor kell megadnunk, ha a megfelelő elérési út nincs a PATH-ban definiálva).

A MAKE abbahagyja a leírófile feldolgozását, ha nem találja valamelyik input file-t, vagy ha egy hívott parancs hibakóddal tér vissza (pl. nem sikerült a fordítás).

9.5 A MAKE opciói

A MAKE futását az alábbi opciókkal módosíthatjuk:

/D	(display date)	Kiírja a megvizsgált file-ok utolsó módosításának dátumát
/I	(ignore)	Figyelmen kívül hagyja az egyes parancsok visszaadott hibakódját, és folytatja a végrehajtást.
/N	(not execute)	Nem hajtja végre az egyes parancsokat, csupán kiírja azokat a képernyőre.
/S	(silently)	A szükséges parancsokat végrehajtja, de azok nem jelennek meg a képernyőn.

9.6 Makro definíciók

A makrok lehetővé teszik, hogy egy konkrét nevet egy szimbolikus névvel (formális paraméterrel) helyettesíthesünk. Ezzel egy adott leíró-file több célra is használható (ugyanazokat a műveleteket más file-okon is végrehajthatjuk) a leírófile újraírása nélkül.

A makródefiníció formája:

szimbolikus-név = konkrét-név

Hivatkozás a makróra:

$(*szimbolikus-név*)

Az ilyen hívások helyére a MAKE a konkrét nevet helyettesíti be. Makrokat a következő helyeken definiálhatunk:

1. A parancssorban, a MAKE hívásakor;
2. A leíró file-ban;
3. A DOS-ban beállított környezetben (SET).

Amennyiben egy definíció több helyen is előfordul a fenti sorrend szerint előbb álló lesz az érvényes. A MAKE három speciális makrót is felismer:

$*   az output file kiterjesztés nélküli nevét jelenti;
$@   az output file kiterjesztés nélküli nevét jelenti;
$**   az input file-ok teljes listáját jelenti.

9.7 Példa a MAKE használatára

A PROG.EXE nevű programunk az alábbi forrásokra épül:

PROG.ASM, P1.ASM, P2.ASM

A leíró-file az alábbi sorokat tartalmazza:

```
 PROG.OBJ :  PROG.ASM  # PROG.OBJ  f gg PROG.ASM-t¢l
       MASM  PROG;

 P1.OBJ   :  P1.ASM
       MASM  P1;
 P2.OBJ   :  P2.ASM
       MASM  P2;
       LIB   PROG.LIB -+P2;
PROG.EXE  :  PROG.OBJ  P1.OBJ  PROG.LIB
       LINK  PROG+P1,,,PROG;
```

A fenti sorok hatására a MAKE elkészíti a futtatható (PROG.EXE) file-t, ha a források bármelyikét módosítjuk, vagy ha a PROG.LIB könyvtár más file-ok miatt megváltozott.

10. Szimbolikus nyomkövető és hibakereső program (SYMDEB)

10.1 Bevezetés

A Microsoft SYMbolic DEBug utility (szimbolikus hibakereső) program végrehajtható (.EXE vagy .COM) file-ok (és bármilyen más bináris file) módosítására, tesztelésére alkalmas. Visszafordítja a programkódot, módosíthatók a regiszterek, töréspontokat tudunk elhelyezni a programban és lépésenként futtathatjuk a programot. A SYMDEB-bel tesztelhetünk minden olyan programot, amelyek a Microsoft nyelvek konvenciói szerinti lebegőpontos emulációt használják.

A SYMDEB minden olyan funkciót, parancsot ismer, amelyet a DEBUG hibakeresőt használó programozók megszoktak. Legjelentősebb újítása az, hogy a forrásprogram objektumaira szimbólumaikkal hivatkozhatunk. Adatokat ellenőrizhetünk és módosíthatunk, eljárásokat hívhatunk meg úgy hogy nem kell tudnunk hogy hol (milyen fizikai címen) vannak a memóriában. Ehhez a szimbólumtáblát tartalmazó (.SYM) file-t kell előállítanunk a szerkesztőprogrammal előállított .MAP file-ból. Erre szolgál a SYMDEB-hez tartozó MAPSYM nevű segédprogram.

A SYMDEB-bel tartós (állandó) töréspontokat helyezhetünk el a programban. Ezeken áthaladva a programfutás mindig megszakad és ellenőrizhetünk, módosíthatunk adatokat és a programot is, lépésenkénti végrehajtással futtathatunk tovább... stb. A töréspontok kikapcsolhatók, törölhetők és újra visszakapcsolhatók.

A SYMDEB kifejezéskiértékelője használni tudja a szimbólumokat és számokat is. Ismeri az aritmetikai és logikai alapműveleteket, hivatkozhatunk az objektumok szegmens- és offset-címére, portról beolvasott adatokra... stb.

A Microsoft Pascal, C és FORTRAN fordító forrás-szintű hibakeresésére is alkalmas a SYMDEB. Kijelzi a lefordított program gépi kódját és a forrássorokat is. A kétféle kijelzési módot együtt és külön-külön is használhatjuk.

A Microsoft cég legújabb fordítóihoz új hibakereső és nyomkövető programot, az ún. CODEVIEW-t készített. Ez lényegesen többet tud a SYMDEB-nél (pl. teljes képernyő használata). Jelenleg a Microsoft 'C' 4.0-ás, FORTRAN 4.0-ás és Quick BASIC 3.0-ás verziójú fordítókhoz adják.

10.2 A SYMDEB indítása

A SYMDEB-et a következő alakú DOS parancs-sorral indíthatjuk el:

SYMDEB [/IBM] [*symfiles*] [[*filespec*] [*args*]]

/IBM	az IBM kompatibilitást jelzi
symfiles	a .SYM szimbólumtáblát tartalmazó file-ok neveit adhatjuk meg
filespec	a végrehajtható program- vagy bináris file neve
args	a végrehajtható program argumentumai, amit a SYMDEB úgy helyez el, mintha a DOS-ból indítottuk volna a programot.

Ha paraméterek nélkül indítjuk a SYMDEB-et, betöltődés után kiírja a bejelentkező üzenetet és a processzor típusát (a 8088-at és a 8086-ot nem tudja megkülönböztetni, IBM XT-n is 8086-ot ír ki).

10.2.1 Indítás program file-lal

Amikor a DOS parancssorban megadjuk egy .EXE vagy egy .COM file nevét is, pl:

```
SYMDEB FILE.EXE
```

a SYMDEB létrehozza a program szegmens prefixet (PSPA) és betölti a programot a rendelkezésre álló legkisebb címû szegmensbe. A file mérete (byte-okban) a BX:CX regiszterpárba kerül és a többi regiszter értéke beállítódik a programfile-nak megfelelően.

10.2.2 Indítás szimbólumokkal

A DOS parancssorban megadhatjuk a szimbólum-file(-ok) nevét is. Ekkor a szimbólumtábla is betöltődik. A szimbólum-file-t a MAPSYM segédprogrammal kell elkészíteni a szerkesztés után, használatát a fejezet 2.5 pontja ismerteti.

Példa:

```
SYMDEB FILE.SYM FILE.EXE
```

10.2.3 Paraméterátadás a betöltendő programnak

A DOS parancssorban megadhatunk paramétereket a futtatandó program számára, pl:

```
SYMDEB FILE.SYM FILE.EXE X.DAT/K
```

Ekkor az .EXE (vagy .COM) file neve mögötti karaktereket a SYMDEB elhelyezi a program PSPA-ban úgy, mintha a DOS indította volna a programfile-t.

A program fejlécről további információkat talál a fejezet 5.26. pontjában a NAME parancsnál.

10.2.4 Indítás file nélkül

Amikor a SYMDEB-et paraméterek nélkül indítjuk, létrehozza a PSPA-t, de nem tölt be programot, ezt a LOAD és NAME parancsok segítségével kell megtennünk, ha szükséges. Paraméter nélküli indítás esetén a szegmensregiszterek a szabad memória legkisebb címére (a PSPA-ra) állítódnak be, az IP értéke 0100H lesz és a többi regiszter nullára áll.

10.2.5 Szimbólum file készítése (MAPSYM)

A szimbólikus hibakeresést biztosító funkciók használatához el kell készítenünk a .SYM file-t. Ennek előfeltétele, hogy a szerkesztőprogramot a /MAP kapcsolóval indítsuk el, vagyis létrehozzuk a .MAP file-t. Ha a forrásszintû nyomkövetést is használni szeretnénk a /LINENUMBERS kapcsolót is meg kell adni. Az így létrehozott .MAP file-ból készít a MAPSYM segédprogram .SYM file-t a SYMDEB számára.

A parancs formája:

MAPSYM [-l|/l]*filespec*

filespec	a .MAP file neve. Ha nem adunk meg kiterjesztést a.MAP az alapértelmezés.
-l|/l	a kétféle megadás egyenértékű és segítségével előírhatjuk, hogy a MAPSYM a konverzió folyamatáról információkat írjon ki a képernyőre

10.3 A kontroll karakterek funkciója

A DOS által biztosított szerkesztési lehetőségek a SYMDEB parancssorában is rendelkezésre állnak.

10.3.1 Parancs megszakítása

Bármelyik SYMDEB parancs végrehajtását megszakíthatjuk a <Ctrl-Break> illetve <CTRL-C> gombok megnyomásával, ekkor visszakapjuk a SYMDEB promptját. A GO paranccsal elindított program csak akkor szakítható meg a <Ctrl-Break>-kel, ha az futás közben ír a képernyőre, vagy adatot kér a billentyűzetről.

Ha a Redirection paranccsal (35. pont) más eszközről vezéreljük a SYMDEB-et, a <CTRL-C> funkció nem működik.

10.3.2 Parancs felfüggesztése

A <CTRL-S> gomb megnyomásával megállíthatunk és továbbengedhetünk minden olyan SYMDEB parancsot, amelyik a képernyőre ír. Ha a Redirection paranccsal (5.31. pont) más eszközről vezéreljük a SYMDEB-et, a <Ctrl-S> hatástalan.

10.3.3 Az NMI használata

Az NMI segítségével bármikor megszakíthatjuk a program futását (végtelen hurok!), de ehhez HW segédeszközre van szükség (pl: IBM Professional Debug Facility). IBM AT-n a System Request segítségével szimulálható az NMI, de ez nem hatásos, ha a program letiltja a megszakításokat.

10.4 A parancsok használata

10.4.1 A parancsok általános felépítése

parancsnév [*paraméter,,,*]

A *parancsnevek* egy vagy két karakteresek, a *paraméterek* a parancs jellegétől függően címkifejezések, karakterek stb. lehetnek. A kis- és nagybetűk közt a SYMDEB nem tesz különbséget. Ha több paramétert adunk meg, elválasztásuk vesszővel vagy tetszőleges számú szóközzel történhet.

10.4.2 Szimbólumok

A szimbólumok karakterekkel vagy aláhúzással, dollárral, Ő jellel vagy kérdőjellel kezdődhetnek és a szimbólumon belül számjegyeket is tartalmazhatnak. Szimbólum lehet bármelyik regiszter vagy flag neve és a .SYM file-ban szereplő globális szimbólum neve. A szimbólumokban a SYMDEB a kis- és nagybetűket nem különbözteti meg. Ha két azonos nevű szimbólum van a .SYM file-ban, csak az először előfordulót tudjuk elérni. Ha valamelyik szimbólum neve azonos egy regiszter vagy flag nevével, akkor azt a szimbólumot nem tudjuk elérni.

10.4.3 Számok

számjegyek[*r*]

A SYMDEB parancsokban a számokat a fenti alakban adhatjuk meg és egész típusú értéket kell adniuk. A *számjegyek*nek meg kell felelniük az [*r*]-rel megadott számrendszernek. Ha [*r*]-t elhagyjuk, hexadecimális az alapértelmezés.

[r] jelentése:

Y - bináris
O,Q - oktális
T - decimális
H - hexadecimális

A megadott szám értéke az [0,65535] tartományba eshet. Ha nagyobb, a SYMDEB annyi jegyet hagy el a szám elejéről, hogy teljesítse ezt a feltételt. A vezető nullák száma nem korlátozott.

10.4.4 Címek

[*szegmens*:]*offset*

A címek a fenti alakban adhatók meg. Bármelyik cím lehet kifejezés is, amely szimbólumokat is tartalmazhat. *Ha a szegmenscímet elhagyjuk a SYMDEB az A,G,L,PT,T,U és W parancsokban a CS, minden más parancsban a DS regiszter értékét használja szegmenscímnek.*

10.4.5 Címtartomány

startcím [*végcím* | L *hossz*]

Címtartomány megadásakor a fenti forma használható. Megadható a tartomány vége vagy a hossza, de a második rész el is maradhat. A *startcím* és *végcím* a 10.4.4 pontnak megfelelően adható meg. Ha a második részt elhagyjuk, a SYMDEB L80H-t helyettesít be.

10.4.6 Sorszámok

.+ |- [*szám*]
.[*filenév*] *szám*
szimólum [+ |- *szám*]

A sorszámok a fenti formák egyikében adhatók meg és a forrásszöveg sorszámát jelentik.

Az első forma relatív sorszámot ad meg az aktuális sorhoz viszonyítva. Ha a *szám* elmarad, a + jel a file utolsó sorát, a - jel az első sorát jelenti. A második forma abszolút sorszámot jelent. Ha a *filenevet* megadjuk, abban a forrásfile-ban, ha nem akkor a CS:IP-nek megfelelő file-ban. A harmadik forma a forrásszöveg *szimbólum*mal megadott objektumát, illetve az ehhez viszonyított relatív sorszámot jelenti. Ha *filenév* vagy *szám* nem létezik, hibajelzést kapunk.

10.4.7 Karaktersorozatok

'[*karakter(ek)*]'
"[*karakter(ek)*]"

Karaktersorozatok a fenti formákban adhatók meg. Ha a sorozat a határoló jelet is tartalmazza, azt kettőzve kell megadni.

10.4.8 Kifejezések

Minden kifejezés értéke egy 8, 16 vagy 32 bites érték. Kifejezés bármelyik parancs argumentumaként használható, ahol 8, 16 vagy 32 bites értéket kell megadni. Kifejezésekben számok, szimbólumok, címek és operátorok szerepelhetnek.

A használható operátorokat az alábbi táblázatok tartalmazzák, a felsorolás sorrendje egyúttal precedenciát is jelent. Az előbb felsorolt operátor precedenciája nagyobb.

Egyoperandusú operátorok

+	pozitív előjel
–	negatív előjel
NOT	egyes komplemens
SEG	az operandus szegmenscíme
OFF	az operandus eltolási címe
BY	a kifejezés kisebb helyiértékű byte-ja
WO	a kifejezés kisebb helyiértékű szava
DW	a kifejezés duplaszó alakja
POI	a kifejezés értékének megfelelő 4 byte-os mutató
PORT	a kifejezésnek megfelelő című portról beolvasott byte
WPORT	a kifejezésnek megfelelő című portról beolvasott szó

Kétoperandusú operátorok

*	szorzás
/	egész típusú osztás (DIV)
MOD	maradék
:	szegmens módosítás
+	összeadás
-	kivonás
AND	bitenkénti "ÉS" művelet
XOR	bitenkénti "KIZÁRÓ VAGY" művelet
OR	bitenkénti VAGY művelet

Azonos precedencia esetén a kiértékelés balról-jobbra történik. A precedencia zárójelekkel felülbírálható.

10.5 A SYMDEB parancsai

10.5.1 Az Assemble (fordítás) parancs

A [*cím*]

A megadott *címtől* kezdődően **befordítja a 8086/8087 processzorok mnemonikjaival megadott utasításokat**. Ha nem adunk meg *címet*, a fordítás a CS:IP értéktől kezdődik.

Minden assembly utasítást az ENTER-rel kell lezárni. Kilépni a parancsból úgy lehet, hogy assembly utasítás helyett üres ENTER-t adunk be.

Az utasítások beírásakor a következő szabályokat kell betartanunk:

- A prefix utasításokat külön sorban kell megadni önálló utasításként.
- A FAR RETURN mnemonikja RETF.
- A stringmozgató utasításokban a mnemonikkal kell megadni az elemek méretét, pl: MVSW, LODSB.
- Az ugrások automatikusan SHORT, NEAR, FAR típusúak lesznek, de ez felülírható a NE[AR] vagy a FAR mnemonikkal.
- A memória hivatkozásoknál a
  WO[RD] PTR és a
  BY[TE] PTR-t
  kell használni, ha az utasításokból ez nem egyértelmű.
- A zárójel nélküli operandusokat immediate (közvetlen) adatnak, a szögletes zárójelbe tetteket memóriahivatkozásnak tekinti a SYMDEB.
- A DB, és DDW direktívák az assemblernek megfelelő módon használhatók.
- Az assemblerben használt minden indexelési írásmód megengedett.
- A mnemonikok szinonímái használhatók, pl: JZ, JE.
- Az FWAIT utasítást külön sorban az előtt az utasítás előtt kell megadni, amelyikre vonatkozik.

Ha szintaktikai hibát vétünk, a SYMDEB az "Error" üzenetet írja ki, és nem fordítja be az utasítást.

10.5.2 A Break Point Set (töréspont megadása) parancs

BP[*n*] *cím* [*m*]

A fenti parancs **állandó töréspontot** helyez el a programban. Egyidőben 10 töréspont lehetséges. Ezek sorszámát jelenti az *n* paraméter. Ha nem adjuk meg, a következő üres töréspont automatikusan hozzárendelődik. **BP** és *n* között **nem lehet szóköz**! A megadott töréspont áthaladás után sem törlődik, mint a GO paranccsal megadott töréspont. Ha az *m* paramétert megadjuk, az első *m* áthaladáskor a töréspont figyelmen kívül marad (*m* egy 16 bites érték). *Az érvényes töréspont elérésekor a SYMDEB kiírja a regiszterek tartalmát,* a flag-eket és az utasítást, és a prompttal jelentkezik. Ekkor bármilyen SYMDEB parancsot kiadhatunk.

Minden töréspont ki- és bekapcsolható, törölhető és felülírható.

10.5.3 A Breakpoint Clear (töréspont törlése) parancs

BC *lista* | *

Törli a *listá*ban felsorolt sorszámú töréspontokat, illetve az összeset * megadása esetén.

10.5.4 A Breakpoint Disable (töréspont kikapcsolása) parancs

BD *lista* | *

Kikapcsolja a *listá*ban felsorolt sorszámú, illetve az összes töréspontot (* megadása esetén). Ezek a BE paranccsal kapcsolhatók vissza.

10.5.5 A Breakpoint Enable (töréspont bekapcsolása) parancs

BE *lista* | *

Bekapcsolja a *listá*ban felsorolt sorszámú, illetve a * megadása esetén az összes töréspontot.

10.5.6 A Breakpoint List (töréspontok kiírása) parancs

BL *lista* | *

Kiírja a *listá*ban felsorolt, vagy az összes töréspontokat a következő formában:

n s cím [m1(m2)]

n	a töréspont sorszáma
s	a töréspont állapota e - bekapcsolva d - kikapcsolva
cím	a töréspont címe. Ha van szimbóluma akkor a szimbólummal, ha nincs, számokkal
m1	a töréspont aktivizálódásáig hátralévő áthaladások száma
m2	a töréspont megadásakor specifikált áthaladási szám

m1 és m2 csak akkor szerepel, ha a BP parancsban megadtuk az *m* paramétert.

10.5.7 A Compare (összehasonlítás) parancs

C *címtartomány cím*

összehasonlítja a *címtartomány* terület tartalmát a *címen* kezdődő terület tartalmával. Ha nem talál eltérést, nem ír ki semmit, illetve ahol eltérés van, kiírja a címeket és a két értéket.

10.5.8 A Display (?) (kifejezés értékének kiírása) parancs

? *kifejezés*

Kiírja a *kifejezés* értékét minden értelmes formában (16 bites, 32 bites hexadecimális, decimális (zárójelben), stringként (idézőjelek közt). A *kifejezés* a 10.4.8 pont szerinti elemekből állhat.

10.5.9 A Dump ASCII (karakteres kiíratás) parancs

DA [*címtartomány*]

A megadott *címtartomány*nak megfelelő memóriatartományt ASCII karakterekként értelmezve kiírja a képernyőre. A vezérlő-karakterek helyett pontot ír. A kiírást az első 0 értékű byte-nál abbahagyja. Ha paramétert nem adunk meg, az előző DA végrehajtásának folytatásaként végzi a kiírást az első 0 értékű byte-ig, de legfeljebb 128 byte-ot ír ki.

10.5.10 A Dump Bytes (byte-os kiíratás) parancs

DB [*címtartomány*]

A megadott *címtartomány* tartalmát hexadecimális és ASCII formátumban kiírja. Az ASCII mezőben a vezérlő-karakterek helyett pontot ír.

10.5.11 A Dump Words (szavas kiíratás) parancs

DW [*címtartomány*]

A megadott címtartomány tartalmát szavakként (az alsó-felső byte-os tárolást átalakítva) kiírja.

10.5.12 A Dump Double words (duplaszavas kiíratás) parancs

DD [*címtartomány*]

A megadott címtartomány tartalmát duplaszóként (szegmens:offset) kiírja.

10.5.13 A Dump Short Reals (valós számos kiíratás) parancs

DS [*címtartomány*]

A megadott *címtartomány* tartalmát rövid valós számokként értelmezve kiírja. Előbb a 4 byte hexadecimális értékét, majd a decimális megfelelőjét írja minden sorba. A decimális számot 16 jegyre írja ki, de ebből természetesen csak az első 7 jegy értékes.

10.5.14 A Dump Long Reals (dupla pontosságú kiíratás) parancs

DL [*címtartomány*]

A megadott *címtartomány* tartalmát hosszú valós számokként értelmezve kiírja. Előbb a 8 byte hexadecimális értékét, majd a decimális megfelelőjét írja ki soronként.

10.5.15 A Dump Ten-Byte Reals (tíz byte-os pontosságú kiíratás) parancs

DT [*címtartomány*]

A megadott *címtartomány* tartalmát 10 byte-os valós számként értelmezve kiírja. Előbb a 10 byte hexadecimális értékét, majd a decimális megfelelőjét írja ki soronként.

10.5.16 A Dump (kiírás folytatása) parancs

D [*címtartomány*]

Az előzőleg használt Dump parancs szerinti formátumban kiírja a *címtartomány*ba eső byte-okat. **A D karakter után legalább egy szóköz megadása kötelező.**

10.5.17 Az Enter (adatbeviteli) parancs

E *cím* [*lista*]

A *cím*mel kezdődő területre **beírja a** *listá***ban megadott byte-okat.**

Ha a *listát* nem adjuk meg, kiírja a <cím> tartalmát és egy pontot, majd vár. Ekkor a következő lehetőségeink vannak:

- Beírunk egy hexadecimális számot, majd az ENTER-t; ekkor felülírtuk a byte-ot.
- Megnyomjuk a szóköz billentyűt: a byte értéke nem változik, és a következő byte-ra lépünk.
- A - jelet írjuk be, ekkor az előző byte-ra lépünk vissza.
- <ENTER>-rel befejezzük az ENTER parancsot.

10.5.18 eXamine Symbol Map (szimbol tábla kiíratása) parancs

X [*]
X ?[*filenév*!][*szegmens*:][*szimbólum*]

A parancs kiírja a betöltött szimbólumokat és címüket. Az X parancs paraméter nélkül kiírja az aktuális szegmens nevét. Az X* alak kiírja az összes szegmens minden szimbólumának nevét és címét. Az X? alak kiírja a megadott szegmensben lévő szimbólum címét. A szegmens és szimbólum megadásakor használhatjuk a '*' karaktert, ez a név hátralévő karaktereit helyettesíti, és kiír minden olyan nevet és címet, ami megfelel az ilyen megadásnak.

10.5.19 A Fill (memória feltöltése) parancs

F *címtartomány lista*

A *címtartománnyal* megadott területet feltölti a *listá*ban megadott tartalommal. A *lista* kevesebb elemet is tartalmazhat, ekkor a teljes listát ismétli a feltöltendő terület végéig.

10.5.20 A Go (program indítása) parancs

G[=*startcím*] [*töréspont*[,,,]]

Átadja a vezérlést a *startcím*en kezdődő gépi kódú programnak. A program vagy végigfut, vagy a *töréspont*ok valamelyikén vagy a BP paranccsal megadott töréspontok valamelyikén megáll. Ezután a vezérlés visszakerül a SYMDEB-hez. Maximum tíz *töréspont* adható meg és ha valamelyiket eléri a program, valamennyi töréspont törlődik. Ha a *startcím*et nem adjuk meg, a vezérlés a CS:IP-nek megfelelő címre adódik át.

A Go parancs úgy indítja el a felhasználó programját, hogy a verembe menti a flag-eket és a CS, IP regisztereket majd egy IRET utasítást hajt végre, ezért a veremben legalább 6 byte-nyi helynek kell lenni a Go parancs kiadásakor.

A töréspontokra a SYMDEB egy-egy INT 3 utasítást ír be (a régi kódot előbb kimenti), és ha valamelyik töréspontra ráfut a program, az összes ideiglenes töréspontot törli, és visszaállítja a régi értékeket.

Ha a program végigfut, a SYMDEB a

```
Program terminated normally
```

üzenetet írja ki.

10.5.21 A Help (?) (segélykérés) parancs

```
?
```

Kiírja a képernyőre a következőket:

```
-?
A [<address>] - assemble
BC[<bp>] - clear breakpoint(s)
BD[<bp>] - disable breakpoint(s)
BE[<bp>] - enable breakpoint(s)
BL[<bp>] - list breakpoint(s)
BP<bp> <address> - set breakpoint
C <range><address> - compare
D[type][<range>] - dump memory
E[type]<address>[<list>] - enter
F <range><list> - fill
G [=<address>[<address>...]] - go
H <value><value> - hexadd
I <value> - input from port
K [<value>] - stack trace
L [<addr>[<drive><rec><rec>]] - load
M <range><address> - move
N <filename>[<filename>...] - name
O <value><byte> - output to port
P [=<address>][<value>] - program step
Q - quit
R [<req>][[=]<value>] - register
S <range><list> - search
S {-P&P+} - source level debugging
```

```
T [=<address>][<value>] - trace
U [<range>] - unassemble
V [<range>] - view source lines
W [<address>[<drive><rec><rec>]] - write
X [?]<symbol> - examine symbols(s)
XO<symbol> - open map/segment
Z <symbol><value>

? <expr> - display expression
! [dos command] - shell escape
. - display current source line
\ - screen flip
> } <device/file> - Redirect output
< { <device/file> - Redirect input
=   <device/file> - Redirect both
* <string> - comment

<expr> ops: + - * / : not seg off by wo dw poi port wport mod
and xor or
<type> : Byte, Word, Doubleword, Asciz, Shortreal, Longreal,
Tenbytereal
```

Ez tartalmazza valamennyi parancs rövid ismertetését.

10.5.22 A Hex (hexadecimális műveletvégzés) parancs

H *kif1 kif2*

Kiírja a *kif1* és *kif2* kifejezések összegét és különbségét hexadecimális formában.

10.5.23 Az Input (byte beolvasása) parancs

I *portcím*

Beolvas egy byte-ot a *portcím*ről és kiírja a képernyőre.

10.5.24 A Load (file betöltése) parancs

L [*cím* [*meghajtó rekord szám*]]

Betölt egy file-t, vagy a lemez megadott logikai szektorától kezdve az előírt számú szektort a megadott *cím*re. Betöltés után BX:CX a betöltött byte-ok számát tartalmazza.

Ha file-t akarunk betölteni, azt előzőleg a Name paranccsal kell megadni. Ha nem adtuk meg, a SYMDEB a DS:005CH címen lévő karaktereket értelmezi filenévnek.

Ha *címet* megadtunk, a betöltés oda történik, ha nem, a CS:0100H címre. Ha logikai rekordokat akarunk betölteni, meg kell adni a *meghajtó kezdőrekord rekordszám* paramétereket. A SYMDEM az adott meghajtóban levő lemez vagy fix lemez DOS partíciójából olvassa be a szektorokat. A legelső szektor 0. sorszámú.

10.5.25 A Move (memórlacím másolása) parancs

M *címtartomány cím*

A *címtartománnyal* megadott területet a *címen* kezdődő területre másolja. Ha a területek átfedik egymást, a másolás akkor is korrekt.

10.5.26 A Name (file-név megadása) parancs

N [*filenév*][*argumentum*]

Megadhatjuk a következő Load és Write parancs számára a file nevét. Az *argumentum* karakterek az elindítandó program argumentumai lesznek, mintha a DOS parancssorban adtuk volna meg. Vigyázat! *A SYMDEB a nevet a DS:5CH az argumentumo(ka)t a DS:80H címen helyezi el.*

10.5.27 Az Open Map (XO) (.SYM file megadása) parancs

XO [*filenév!*][*szegmensnév*]

Kijelöli az aktív .SYM file-t és/vagy szegmenst.

10.5.28 Az Output (byte kiírása) parancs

O *portcím byte*

A parancs a megadott *byte*-ot kiírja a *portcím* I/O portra.

10.5.29 A PTrace (gyorsított programvégrehajtás) parancs

P [=*startcím*][*szám*]

Végrehajt *szám* darab gépi kódú utasítást *startcím*től kezdve. Ha a *startcím*et elhagyjuk, CS:IP az alapértelmezés. *szám* elhagyása esetén egy utasítást hajt végre.

Minden utasítás végrehajtása után kiírja a regiszterek és flag-ek értékét, ahogy a T parancs is teszi. Ez az utasítás folyamatosan (nem lépésenként) hajtja végre a hívott szubrutinokat és a software megszakításokat.

(Szegmensregisztert töltő utasításoknál: MOV/POP DS/ES/SS, LDS, LES a következő utasítást is végrehajtja!)

10.5.30 A Quit (vége) parancs

Q

Visszatér a DOS-hoz, befejezve a SYMDEB működését.

10.5.31 Az átirányítási <, >, = parancsok

< *eszköznév*
> *eszköznév*
= *eszköznév*

A DOS valamelyik logikai eszközére kapcsolja át az input és/vagy outputokat, de ez csak a SYMDEB által képernyőre írt, illetve beolvasott adatokra vonatkozik. Olyan programok teszteléseskor hasznos, amelyek megtervezett képernyőre (grafikus képernyőre) dolgoznak.

Az *eszköznév* bármelyik logikai eszköz lehet, általában a COM1 használható erre a célra. Visszakapcsolni a konzolra az =CON paranccsal lehet.

10.5.32 A regiszter parancs

R [*regnév* [*érték*]]

Megnézhetjük és módosíthatjuk a CPU regisztereinek és a flag-eknek az értékét. Ha nem adunk meg regiszter nevet, kiírja a teljes regiszterkészlet állapotát. Ha csak nevet adunk meg, értéket nem, kiírja a régi értéket és várja, hogy megadjuk az újat. Ha értéket is adunk, az a megadott regiszterbe kerül.

A következő regiszternevek használhatók:

AX BX CX DX SP
BP SI DI DS ES
SS CS IP PC F

IP és PC egyaránt az utasítás-számlálót jelenti, F a flag-eket. A flag-ek értékeire a következő elnevezések adhatók meg:

flag	1	0
overflow	OV	NV
direction	DN	UP
ITenable	EI	DI
sign	NG	PL
zero	ZR	NZ
aux carry	AC	NA
parity	PE	PO
carry	CY	NC

10.5.33 A Search (keresés) parancs

S *címtartomány lista*

A megadott *címtartomány*ban megkeresi a *listá*ban megadott minta előfordulásait. A megtalált minták címét kiírja. A mintát megadhatjuk hexadecimális és (idézőjelek közt) karakteres formában is.

10.5.34 A Set Source Mode (forrássorok kiírása) parancs

S [-| + |&]

A Microsoft Pascal, C és FORTRAN használata esetén a forrássorok kiírását kapcsolhatjuk ki és be ezzel a paranccsal.

Az S+ bekapcsolja, az S- kikapcsolja a forrássorok kiírását. Az S& formájú parancs után mind a forrássorokat, mind a visszafordított gépi kódot kiírja.

10.5.35 A Trace (programindítás nyomkövetéssel) parancs

T [=*startcím*] [*szám*]

Működése azonos a P paranccsal (5.29 pont), de ez az utasítás a szubrutinokat és a software megszakításokat is lépésenként hajtja végre.

10.5.36 Az Unassemble (visszafordítás) parancs

U [*címtartomány*]

Kilistázza a *címtartományba* eső gépi kódú programot, visszfordítva mnemonikokra az utasításokat. Az S paranccsal beállított üzemmódnak megfelelően a forrássorokat és a visszafordított assembly programot egyaránt ki tudja írni.

10.5.37 A Write (file kiírása) parancs

W [*cím* [*meghajtó rekord szám*]]

Kiír a lemezre *cím*től kezdve BX:CX darab byte-ot az N paranccsal megadott file-ba, illetve a *meghajtó rekord szám* paraméterek megadása esetén a lemez *rekord* logikai szektorától kezdve *szám* darab rekordot ír fel.

10.6. Hibaüzenetek

Ha egy parancsot hibásan adunk meg, a következő sorban egy felfelé mutató nyíllal (|) megjelöli azt a szintatikai egységet, amit már nem tudott értelmezni, és kiírja az "Error" szöveget.

Egyéb hibák esetén a parancs végrehajtása félbeszakad és a következő hibaüzeneteket írja ki a SYMDEB (és újabb parancsra vár):

Bad flag:	Hibás flag. A flag nevét hibásan adtuk meg a parancsban.
Too many breakpoints!	Túl sok töréspont. A Go parancsban több, mint 10 töréspontot adtunk meg.
Bad register!	Az R parancsban hibásan adtuk meg egy regiszter nevét
Double flag!	Az R parancsban egy flag-hez két értéket próbáltunk megadni egyszerre.
Breakpoint list or * expected!	A BC, BD, BL parancsban a töréspontok listáját vagy a *-ot nem adtuk meg.
Error reading .SYM file!	Hiba a SYM file olvasásakor. A megadott .SYM file-t nem tudja beolvasni a SYMDEB. A file üres, vagy hibás.

10.7. A SYMDEB kompatibilis fordítók

A következő assemblerek és fordítók képesek SYMDEB kompatibilis kódot előállítani (melyekből .SYM generálható)

Microsoft

FORTRAN 3.0 vagy nagyobb
PASCAL 3.0 vagy nagyobb
C 2.0 vagy nagyobb
MASM 1.0 vagy nagyobb
BASIC Compiler 1.0 vagy nagyobb
Business BASIC Compiler 1.0 vagy nagyobb

IBM

FORTRAN 2.0 vagy nagyobb
Pascal 2.0 vagy nagyobb
Macro Assembler 1.0 vagy nagyobb
BASIC Compiler 1.0 vagy nagyobb

A forrássorok kiírásához szükséges információkat automatikusan generálják:

Microsoft

FORTRAN 3.0 vagy nagyobb
Pascal 3.0 vagy nagyobb

IBM

FORTRAN 2.0 vagy nagyobb
Pascal 2.0 vagy nagyobb

A Microsoft C fordítónál kapcsolóval kell előírni a forrásszintű hibakereséshez szükséges információk generálását.

11. Keresztihivatkozási-lista készítő program (CREF)

11.1 Bevezetés

A CREF program abc-sorrendben felsorolja a programban használt összes szimbólumot és felsorolja összes előfordulásának helyét. Ennek segítségével gyorsíthatjuk a hibakeresést és könnyen megtalálhatók a deklarált, de nem használt szimbólumok, amelyek zavarják a program érthetőségét.

11.2 A CREF használata

A CREF program az assembler által készített .CRF file-t olvasható szövegfile-lá konvertálja. Ehhez természetesen a MASM indításakor a negyedik paramétert (a .CRF file nevét) meg kell adnunk. Pl:

```
MASM prg.asm,prg.obj,prg.lst,prg.crf
MASM prg,,,prg;
MASM
        Source filename [.ASM]: prg
        Object filename [prg.OBJ]: prg
        Source listing [NUL.LST]:
        Cross reference [NUL.CRF]: prg
```

Ezután a CREF-et elindítva elkészíti az olvasható formátumú keresztreferencia file-t.

A parancs formája:

CREF [*crf-file*[,*ref-file*]]

crf-file a MASM által készített .CRF file neve;
ref-file az olvasható formájú keresztihivatkozási lista file neve.

A file neveknél kiterjesztést nem kötelező megadni, az alapértelmezések .CRF és .REF. A névmegadás szabályai azonosak a MASM-nál megszokottal.

Például:

CREF prg; a PRG.CRF file-ból PRG.REF file-t készít

Ha nem adunk meg file-nevet a CREF is rákérdez:

```
CREF
        Cross reference [.CRF]:prg
        Listing [prg.REF]:
```

Ha a CREF futása közben a <Ctrl-Break>-et lenyomjuk a program befejezi működését és nem hoz létre .REF file-t.

11.3 A keresztreferencia-lista

A keresztreferencia-lista névsorban tartalmazza a forrásszövegben definiált összes szimbólumot és utána felsorolja azokat a sorszámokat, ahol a szimbólumra hivatkoztunk. A definíciós sor számát #-kal jelöli meg. Ha egy szimbólum mellett csak ilyen sorszám szerepel, azt csak definiáltuk, de nem használtuk fel, tehát felesleges! Ha a forrásprogramban TITLE direktívával megadtunk fejlécet, azt itt is kiírja a CREF minden lap tetejére.

11.4 Hibaüzenetek

Ha a CREF futás közben hibát talál, kiírja a következő hibaüzenetet és befejezi működését.

Fatal I/O Error:XX
In: file-név

XX a következő lehet:

101	Rossz meghajtónevet adtunk meg, vagy fatális lemezhiba
105	A meghajtó ajtaja nyitva van vagy a printer nincs bekapcsolva
108	A lemez megtelt
110	A megadott .CRF file-t nem találja.
111	A lemez írásvédett, a .REF file-t nem lehet létrehozni.

103, 104, 106, 112-115 CREF belső hibák.

12. Hibajelzések

12.1 Az assembler hibaüzenetei

A MASM külön jelzi a program (assembler) és az I/O hibákat. A hibák két kategóriáját különbözteti meg: a Warning és a Fatal (figyelmeztető és fatális) hibát.

Figyelmeztető hibának ítéli azokat az eseteket, amikor képes volt az adott utasítást értelmezni, de valami félreértésre okot adó eseményt kíván jelezni. Ekkor az OBJ file-t (ha kértük) létrehozza.

Fatális (végzetes) hibát akkor jelez, ha nem tudja az utasítást értelmezni, vagy ellentmondásba ütközik a fordítás során. Ekkor OBJ file nem keletkezik és esetleg a fordítás is félbeszakad.

A fordítás végén a végzetes és a figyelmeztető hibák számát kiírja a képernyőre és az a lista-file-ba is belekerül.

Az újabb változatok a Fatal Error helyett a Severe Error (komoly hiba) kifejezést használják.

12.1.1 Assembler hibák

Programhiba esetén az assembler kiírja a file nevét, a hibás sor számát, egy hibakódot és a hiba megnevezését.

0: **Blokk egymásba skatulyázási hiba**
Egymásba skatulyázható eljárások, szegmensek, adatstruktúrák, makrók nincsenek korrektül lezárva. (Egy blokkot lezártunk, és ennek egy belső blokkja még nyitva van.)

1: **Felesleges karakterek a sorban**
Az utasítást, direktívát sikerült értelmezni, de a sorban még van értékes karakter. (Általában figyelmeztető hiba.)

2: **Már definiált regiszter**
Az assembler belső logikai hibát talált.

3: **Ismeretlen szimbólum típus**

4: **Szimbólum újradefiniálása**

5: **Többszörösen definiált szimbólum**

6: **Fázishiba**
Valamely cimke értéke a fordítás két menetében nem azonos. Tipikus oka hogy egy SHORT-nak deklarált előrehivatkozó ugrás cimkéje távolabb van 127 byte-nál, vagy egy előrehivatkozó ugrás NEAR típusú és a cimke FAR-nak van deklarálva. Az ilyen hiba megkeresésében a /D kapcsoló (listakészítés mindkét menetben) segíthet.

7: **Már van ELSE ág**
Egy újabb ELSE-t akarunk megnyitni, holott az előzőt még nem zártuk le.

8: **Nincs nyitott feltétel**
ELSE vagy ENDIF előzetes IF nélkül.

9: **Nem definiált szimbólum**

10: **Szintatikai hiba**

11: **Típusmegadás itt nem megengedett**

12: **Itt csoportnév szükséges**
Az adott helyen csoport (GROUP) név-nek kell állnia, és a fordító más szimbólumot talált.

13: **Az első menetben ismertnek kell lennie**
Az adott hivatkozásnak már az első menetben ismertnek kell lennie. Pl. egy IF utasítás argumentumának kiszámíthatónak kell lennie az első menetben is.

14: **Nem megengedett szimbólum típus**
Egy PUBLIC szimbólumot nem megengedett helyen használ.

15: **A szimbólum már definiálva van**
Egy szimbólumot más típusúnak kíván újradefiniálni.

16: **Foglalt szó**
Pl: MOV nem lehet változónév

17: **Nem megengedett előrehivatkozás**
Az első menet hibajelzése, csak akkor lép fel, ha mindkét menetben listát kértünk.

18: **Itt csak regiszter állhat**
Az adott helyen szimbólum, vagy kifejezés nem megengedett.

19: **Hibás a regiszter típusa**
Ilyen típusú regiszter itt nem használható. Pl: INC CS

20: **Itt szegmens vagy csoport kell**
Az adott helyen csak szegmens vagy csoport (GROUP) adható meg.

21: **A szimbólumnak nincs szegmense**
A SEG utasítást olyan szimbólumra alkalmaztuk, amelynek nincs szegmenscíme.

```
Pl:   CR  EQU  13
      MOV  AX,seg CR
```

22: **Itt típusmegadás szükséges**
Ezen a helyen csak a WORD, DW, QW, BYTE, TB fordulhat elő.

23: **Már lokálisnak deklarálta**
Egy korábban lokálisnak deklarált szimbólumot EXTERNAL-nak adtunk meg.

24: **A szegmens paraméterei megváltoztak**
Egy ilyen azonosítójú szegmens más paraméterekkel már szerepelt.

25: **Hibás igazítás illetve kombináció megadás**
Paraméterhiba a SEGMENT direktívában.

26: **Többszörösen definiált szimbólumra hivatkozik**

27: **Operandus hiányzik**

28: **Műveleti utasítás (operátor) hiányzik**

29: **Osztás 0-val, vagy túlcsordulás**
A kifejezésben 0-val kellene osztani, vagy a kifejezés értéke túllépi az adott típus értékhatárát.

30: **Léptetés számláló hiba**
A léptetést megadó kifejezés értéke negatív.

31: **Az operandusok eltérő típusúak**
Pl: MOV AX,DL

32: **EXTERNAL itt nem állhat**
Pl: DB M DUP(?) nem megengedett, ha M EXTERNAL

33: **Itt csak rekord mezőnév állhat**

34: **Itt csak rekord vagy mezőnév állhat**

35: **Az operandus méretét meg kell adni**

36: **Itt csak változó, címke vagy konstans állhat**

37: **Itt csak struktúra mezőnév állhat**

38: **A bal operandus csak szegmens lehet**
A ':' operátor bal operandusa csak szegmens- (csoport-) név vagy szegmensregiszter lehet.

39: **Egy operandus csak konstans lehet**
Az összeadó operátor hibás használatára utal.

40: **Az operandusoknak azonosaknak kell lenniük, vagy egyiküknek konstansnak kell lennie.** (Hibás kivonás kijelölés.)

41: **Egyszerű típus-operandust kell használni.**
(Itt nem megengedett a STRUC, FIELDS, NAMES, BYTE, WORD illetve DW, helyettük változót kell használni.)

42: **Itt csak konstans állhat**
Az adott helyen olyan kifejezésnek kell állnia, amely fordítási időben kiértékelhető.
Pl: MOV AX,2+K
ahol K EXTERNAL szimbólum.

43: **Az operandusnak szegmenssel kell rendelkeznie**
A SEG direktíva hibás használata

44: **Itt adatnak kell szerepelnie**
Kódra hivatkozik olyan helyen, ahol adatnak kell állnia.
Pl: MOV AX,kod-cimke

45: **Itt kód-nak kell szerepelnie**
Adatra hivatkozik olyan helyen, ahol kódnak kell állnia.

46: **Már volt bázisregiszter**
Két bázisregiszter nem lehet egy címben

47: **már volt indexregiszter**
Két indexregiszter nem lehet egy címben.

48: **Itt index- vagy bázisregiszter kell.**
A szögletes zárójelben az utasítás index- vagy bázisregisztert vár.

49: **Itt nem állhat regiszter**
Az utasítás szintaxisa szerint itt nem állhat regiszter

50: **Az érték túl nagy**
Pl: MOV AL,256

51: **Az operandus nincs a kódszegmensben**
Az operandus az aktuális kódszegmensből nem érhető el.

52: **Nem megengedett operandus típus**
Az utasítás ilyen operandussal nem létezik

53: **Túl nagy relatív ugrás**
Feltételes ugrás -128 vagy + 127-nél távolabbi cimkére.

54: **Az index kifejezés nem számítható ki**

55: **Hibás regiszter érték**
A megadott regiszter érték nem írható be az op.kód regiszter mezőjébe, mert nagyobb 7-nél.

56: **Itt közvetlen adat nem használható**
Pl: PUSH 2

57: **Itt ilyen méretű elem nem állhat**
Pl: az SHR operátor nem alkalmazható duplaszóra

58: **Itt byte-regiszter nem használható**
Pl: PUSH AL

59: **A CS regiszter itt nem állhat**
Pl: XCHG CS,AX

60: **Itt csak az AX vagy az AL állhat**

61: **Szegmensregiszter hibás használata**
Pl: MOV DS,00FAH

62: **Hiányzik vagy nem elérhető a kódszegmens**
Elérhetetlen cimkére ugrás.

63: **Hibás operanduskombináció**

64: **NEAR JMP vagy CALL másik szegmensbe**

65: **A cimke szegmense nem módosítható**

66: **A prefix után nem op.kód szerepel**
A REP prefix után nem op.kód következik

67: **Az ES szegmens nem módosítható**
Pl: STOS DS:CEL

68: **Az adat nem érhető el**
Az utasításhoz tartozó szegmensregiszterhez nem az adatnak megfelelő szegmenst rendeltük hozzá.

69: **A kód szegmensen kívül van**
Egy kódot generáló utasítás nincs szegmens belsejében.

70: **BYTE típusú szegmensben EVEN nem lehet**

72: **Az ismétlésszám nem pozitív**
A DUP direktíva előtt 0 vagy negatív szám nem állhat.

73: **Már EXTERNAL-nak deklaráltuk**
Egy korábban EXTERNAL-nak megadott szimbólumot lokálisnak deklaráltunk.

74: **A megadott DUP túl nagy**
A DUP-ok egymásba skatulyázása túl sok a szerkesztőprogramnak.

75: **A ? hibás használata**
Pl: DB ? + 5

76: **Túl sok kezdőérték**
Egy rekordhoz vagy struktúrához túl sok kezdőértéket adtunk meg.

77: **Hibás kezdőérték lista**
A struktúra kezdőértékei nincsenek < > zárójelben.

78: **Nem megengedett direktíva a struktúra-deklarációban**
Struktúradeklarációban csak megjegyzés és definiáló direktíva állhat.

79: **Nem megengedett DUP használat**
Egy struktúra inicializáló utasításban hibásan használta a DUP direktívát.

80: **STRUC inicializáló utasításban mező illegális értékadása**

81: **Hibás típus a struktúra kezdőértékben**

82: **Regiszterre nem lehet előre hivatkozni**

83: **EQV direktíva saját magára hivatkozik**
ALFA EQU ALFA

84: **Ez a 8087 utasítás nincs emulálva**

85: **Hibás file-vég**
vagy az END utasítás hiányzik, vagy valamelyik blokkot nem zárta be.

12.1.2 I/O hibák

Az I/O hibaüzenetek formája:

MASM Error – *hibaüzenet*
In: *filenév*

A hibaüzenetek:

101: **Hard data**	Hibás adat
102: **Device name**	Eszköznév hiba
103: **Operation**	Műveletihiba
104: **File system**	File-kezelő hiba
105: **Device offline**	Az eszköz ki van kapcsolva
106: **Lost file**	Sérült file
107: **File name**	File név hiba
108: **Device full**	Az eszköz (lemez) megtelt
109: **Unknown device**	Ismeretlen eszköz
110: **File not found**	A file nincs meg
111: **Protected file**	Írásvédett file
112: **File in use**	A file-t használják
113: **File not open**	A file nincs nyitva
114: **Data format**	Adatformátum hiba
115: **Line too long**	Túl hosszú sor

12.2. A szerkesztő hibaüzenetei

About to generate .EXE file.
Change diskette in drive A: and press ENTER.
Ha a /P kapcsolót megadtuk, mielőtt az .EXE file-t felírná a lemezre, kiírja a fenti üzenetet. Ekkor kell lemezt cserélni és az ENTER gomb lenyomása után az új lemezre írja a szerkesztés eredményét.

Ambiguous switch error: 'X'
Ismeretlen kapcsoló. A parancsban megadott 'X' kapcsolót nem ismeri fel a program. A szerkesztést felfüggeszti.

Array element size mismatch
Tömb elem méret ellentmondás. Egy közös tömb két deklarációja közt ellentmondás van. Assemblyben a 'közös tömb' kategória nem létezik, ezért ez a hiba nem léphet fel.

Attempt to put segment <name> in more than one group in file <filename>
A <name> nevű szegmens két különböző csoporthoz (group) lett hozzárendelve. Korrigálni kell a hibát és újrafordítani a programot.

Bad value for CparMaxAlloc
A /CPARMAXALLOC kapcsolóval megadott érték nincs az [1,65535] intervallumban.

Cannot find library: <filename.LIB>.
Enter new file spec:
A szerkesztő nem találja a <filename> nevű könyvtárat, ezért kéri, hogy a helyette használandót adjuk meg (elérési út és meghajtó is megadható).

Cannot open list file
A directory vagy a lemez megtelt.

Cannot open response file
A megadott válasz-file-t nem találja, hibás file-nevet adtunk meg.

Cannot nest response file
A válasz-file-okat nem lehet egymásbaskatulyázni. (Egy válasz-file-ban másik válasz-file-ra történik hivatkozás).

Cannot open run file
A directory vagy a lemez megtelt

Cannot reopen list file
A felhasználó nem tette vissza a megfelelő diszket, amikor a szerkesztő ezt kérte. A szerkesztést felfüggeszti.

Common area longer than 65536 bytes
A felhasználói program több mint 64K közös változót használ.

Data record too large
A fordító vagy az assembler hibás object modult állított elő. Értesítse a Microsoft céget a garanciajegyen.
(ha van Önnek!)

Dup record too large
A LIDATA record (az object modulban) 512 byte-nál hoszszabb. Egy struktúradefiníció túl komplikált a forrásszövegben. Pl:

ARRAY 10 dup(11 dup(12 dup(13 dup(...)))))
Egyszerűsítsük a struktúrát úgy hogy egy utasítás 512 bytenál kisebb kódot generáljon.

<filename> is not a valid library
A <filename> névvel megadott file nem érvényes (szabályos) könyvtár. A szerkesztést felfüggeszti.

Fixup overflow near <num> in segment <name> in <filename> offset <num>
A hibát a következők idézhetik elő:
1. egy csoport (group) nagyobb 64K-nál.
2. a szerkesztett programban egy CALL SHORT, vagy egy JMP SHORT másik szegmensre hivatkozik (a kódszegmens nagyobb mint 64K)

3. valamelyik cimkéje megegyezik egy könyvtári cimkével
4. egy EXTRN deklaráció valamelyik szegmens belsejében van (tegyük a szegmensen kívülre a deklarációt)

Incorrect DOS version, use DOS 2.0 or later
A LINK csak a 2.0 vagy nagyobb verziószámú DOS-szal képes futni.

Insufficent stack space
Nincs elég memória a program szerkesztéséhez.

Interrupt number exceeds 255
Az /OVERLAYINTERRUPT kapcsolóval megadott szám nagyobb mint 255. Az érvényes tartomány = [4,255].

Invalid numeric switch specification
Hibás szám egy kapcsoló paraméterében.

Invalid object modul
Az object modulok valamelyike hibás, próbálja őket újrafordítani.

NEAR/HUGE conflict
Egy közös változóra eltérő hivatkozások történtek.
Assemblerben ez a hiba nem fordulhat elő.

Nested left parentheses
Az overlay megadása hibás a parancssorban.

No object modules specified
A szerkesztőt object file-név nélkül indítottuk.

Out of space on list file
A lemez megtelt.

Out of space on run file
A lemez, amire a .EXE file kerül, megtelt.

Out of space on scratch file
Az alapértelmezés szerinti meghajtóban lévő lemez megtelt.

Overlay manager symbol already defined/ < name >
Valamelyik cimkéje azonos az overlay kezelő cimkéjével. Cserélje ki a cimkét.

Please replace original diskette in drive:A and press ENTER
Ez az üzenet akkor fordul elő, ha a /P kapcsolót használta, a szerkesztő az .EXE file-t felírta a lemezre és kéri vissza az eredeti lemezt. Tegye vissza a lemezt, és nyomja meg az ENTER billentyűt.

Relocation table overflow
Több mint 16384 távoli hivatkozás (FAR JMP vagy CALL) van a programban. Alakítsa át közeli hivatkozásra az ugrásokat ahol csak lehet, és próbálkozzon újból a szerkesztéssel. Pascal és Fortran használata esetén lehet, hogy elég, ha a debug állapotot kikapcsolja.

Segment limit too high
A szerkesztő nem tudja elhelyezni a táblázatait a memóriában, mert túl sok szegmenst írtunk elő. A /SEGMENT kapcsolóval állítsuk kisebbre a limitet (az alapértelmezés 128), vagy szabadítsunk fel helyet a gépben (SIDEKICK, SUPERKEY,...stb.)

Segment size exceeds 64 K
Valamelyik szegmens mérete nagyobb, mint 64 Kbyte.

Symbol table overflow
A szerkesztett program túl sok PUBLIC, EXTRN, SEGMENT, GROUP... stb. globális szimbólumot használ. Csökkentse ezek számát, vonjon össze szegmenseket, csoportokat... stb.

Terminated by user
CTRL-C-vel megszakította a szerkesztő működését.

Too many external symbols in one module
Valamelyik object modulban túl sok külső (EXTRN) hivatkozás van, csökkentse a számukat.

Too many group-, segment- and classnames in one module
Valamelyik object modulban túl sok csoport, szegmens vagy osztálynevet használ. Csökkentse ezek számát.

Too many groups
A szerkesztett program több, mint 9 csoportot (group) tartalmaz, csökkentse számukat.

Too many GRPDEFs in one module
A szerkesztett program valamelyik modulja több, mint 9 csoportot (group) tartalmaz, csökkentse számukat.

Too many segments
A szerkesztett program túl sok szegmenst tartalmaz. A /SEGMENT kapcsolóval állítsa feljebb a limitet.

Too many segments in one module
Valamelyik object modul több, mint 255 szegmenst tartalmaz.
Csökkentse a szegmensek számát.

Too many TYPEDEFs
Valamelyik object modul túl sok típusdefiníciót tartalmaz.

Unexpected end of file on library
A szerkesztő futása közben kicserélte azt a lemezt, amelyik a könyvtárat tartalmazza.

Unexpected end of file on scratch file
A VM.TMP file-t tartalmazó lemezt kicserélte.

Unmatched left parenthesis
Az overlay megadása hibás a parancssorban.

Unmatched right parenthesis
Az overlay megadása hibás a parancssorban.

Unrecognized switch error
Érvénytelen karakter a parancssorban, valamelyik kapcsoló megadása hibás.

VM.TMP In an illegal file name and has been ignored
Object file névként VM.TMP-t adott meg. Ilyen nevű object modult nem szabad használni. Nevezze át.

Warning: no stack segment
A szerkesztett program nem tartalmaz stack kombináció típusú szegmenst. Ez .COM-nak szánt file esetén nem hiba!

Warning: too many local symbols
Sorbarendezett listát kért a szerkesztőtől, de túl sok a lokális szimbólum, ezért a lista nem lesz sorbarendezve.

Warning: too many public symbols
Sorbarendezett listát kért a public szimbólumokról, de túl sok szimbólum létezik, ezért a lista nem lesz sorbarendezett.